TECHNICAL REPORT RDMR-SS-12-11

METHOD OF CHARACTERISTIC (MOC) NOZZLE FLOWFIELD SOLVER—USER'S GUIDE AND INPUT MANUAL

Kevin D. Kennedy

System Simulation and Development Directorate
Aviation and Missile Research, Development,
and Engineering Center

January 2013

Distribution Statement A: Approved for public release;
Distribution is unlimited.

DESTRUCTION NOTICE

FOR CLASSIFIED DOCUMENTS, FOLLOW THE PROCEDURES IN DoD 5200.22-M, INDUSTRIAL SECURITY MANUAL, SECTION II-19 OR DoD 5200.1-R, INFORMATION SECURITY PROGRAM REGULATION, CHAPTER IX. FOR UNCLASSIFIED, LIMITED DOCUMENTS, DESTROY BY ANY METHOD THAT WILL PREVENT DISCLOSURE OF CONTENTS OR RECONSTRUCTION OF THE DOCUMENT.

DISCLAIMER

THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED BY OTHER AUTHORIZED DOCUMENTS.

TRADE NAMES

USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL ENDORSEMENT OR APPROVAL OF THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 2013	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Method of Characteristic (MOC) Nozzle Flowfield Solver—User's Guide and Input Manual	
6. AUTHOR(S) Kevin D. Kennedy	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Commander, U.S. Army Research, Development, and Engineering Command ATTN: RDMR- SSM-A Redstone Arsenal, AL 35898-5000	8. PERFORMING ORGANIZATION REPORT NUMBER TR-RDMR-SS-12-11
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited.	A

13. ABSTRACT *(Maximum 200 Words)*

The Method of Characteristic (MOC) Nozzle Flowfield Solver code is a structured, Two-Dimensional (2-D), isentropic supersonic flow solver for rocket nozzle problems. The code uses a marching step method based on a second order modified Euler predictor/corrector scheme with iteration. The initial starting line is determined either by a user-input profile or by Sauer's method for determining the sonic line. Initial wall options include a starting circular arc that can be followed by a second order quadratic wall based on user inputs. The remaining downstream wall is then determined by the local characteristic and mass flow rate. A simple boundary-layer correction using a one-eighth power law is used to determine the viscous corrected shape. The software coding is written in Fortran 90. Input files are used to transfer information to the program.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Method of Characteristic (MOC), Nozzle, Flowfield Solver	127
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-**01**-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

TABLE OF CONTENTS

TABLE OF CONTENTS (CONCLUDED)

LIST OF TABLES

1. INTRODUCTION

The Method of Characteristic (MOC) Nozzle Flowfield Solver code is a structured, Two-Dimensional (2-D), isentropic supersonic flow solver for rocket nozzle problems. The code uses a marching step method based on a second order modified Euler predictor/corrector scheme with iteration. The initial starting line is determined either by a user-input profile or by Sauer's method for determining the sonic line. Initial wall options include a starting circular arc that can be followed by a second order quadratic wall based on user inputs. The remaining downstream wall is then determined by the local characteristic and mass flow rate. A simple boundary-layer correction using a one-eighth power law is used to determine the viscous corrected shape. The software coding is written in Fortran 90. Input files are used to transfer information to the program.

2. PROGRAM READS AND WRITES

This section describes the files that are both used and generated by the MOC code. A description of the contents of each file is given in the following section.

2.1 File Description

The input and output files utilized by the code are listed in Table 1. The user must provide the files that are read by the MOC code.

Table 1. Summary of Input/Output Files

Filename	File Descriptor	Format	Program Read/ Writes
Nozzle.inp	Main input variable file	ASCII	Read
Profile.dat	User input start line	ASCII	Read
CENTERLINE.dat	Centerline properties	ASCII	Write
SLOPE.dat	Wall slope file	ASCII	Write
SLOPE_SMOOTHED.dat	Fitted wall slope file	ASCII	Write
WALL.dat	Wall file	ASCII	Write
WALL_BOUNDARY_CORRECTED.dat	Boundary layer wall file	ASCII	Write
WALL_BOUNDARY_CORRECTED_SMOOTHED.dat	Corrected BL wall file	ASCII	Write
WALL_SMOOTHED.dat	Fitted wall file	ASCII	Write
STARTING_CONDITIONS.dem	Start line plot file	GNU	Write
WALL_DEFINITION.dem	Wall shape plot file	GNU	Write
X_Y_FUN.dem	Nozzle flow plot file	GNU	Write
fort.7	Start line solution	ASCII	Write
fort.8	Circular arc solution	ASCII	Write
fort.9	2nd-order wall solution	ASCII	Write
fort.10	Final C+ Chara line	ASCII	Write
fort.11	Turning region flow	ASCII	Write
fort.13	Inviscid flow boundary	ASCII	Write

2.2 Program Reads

This section details the files that are read into the MOC code. The user is required to generate these files.

2.2.1 Nozzle.inp File

This file is the main input file. It is in Fortran Namelist format. Details of this format are given in Section 3.

2.2.2 Profile.dat File

The “Profile.dat” is dependent on user input. It contains the start line properties for the initial valued solution. It is a formatted American Standard Code for Information Interchange (ASCII) file. This file must start with conditions on the centerline and move up to the wall. It must be in the format of I, J, K, X, Y, Z, Rho, U, V, W, and P; where I, J, and K are integer index values; X, Y, and Z are the point locations (Z is not used); Rho is the density; U, V, and W are the velocity components; and P is the pressure. The temperature is calculated by the equation of state from the density and pressure.

2.3 Program Writes

This section details the files that are written out by the MOC code. The main source of output and errors is directed to Fortran Unit 6, which typically is the main terminal widow from which the code was executed. It is left to the user to capture this information.

2.3.1 CENTERLINE.dat File

This file is written out by the MOC code and contains the centerline properties. The radial location (Y), V velocity, and Flow angle are defined to be of zero value. Breaks in this file contain header information.

2.3.2 SLOPE.dat File

This two column formatted ASCII file contains the inviscid wall slope information. It is in the form of axial location verses the wall slope.

2.3.3 SLOPE_SMOOTHED.dat File

This two column formatted ASCII file contains the inviscid wall slope information after it has been processed through the least-squared curve fit approximation. The order of this fit is a user input variable (IORDER) in namelist INPUT.

2.3.4 WALL.dat File

This multi-column formatted ASCII file contains the inviscid wall properties. It contains a header row detailing the wall locations and properties.

2.3.5 WALL_SMOOTHED.dat File

This two-column formatted ASCII file contains the inviscid wall location after it has been processed through the least-squared curve fit approximation.

2.3.6 WALL_BOUNDARY_CORRECTED.dat File

This multi-column formatted ASCII file contains the boundary layer corrected file. This boundary layer model is a simplest model that adds a correction factor to the wall based solely on local conditions and axial length. It contains an I, J index pair; the original inviscid X, Y point location; normalized temperature; viscosity; Reynolds number; a one-eighth power law boundary layer thickness; and the new viscous radial location.

2.3.7 WALL_BOUNDARY_CORRECTED_SMOOTHED.dat File

This two-column formatted ASCII file contains the boundary layer corrected wall after it has been processed through the least-squared curve fit approximation.

2.3.8 STARTING_CONDITIONS.dem File

This plot file is in the GNU plot format. GNU is a portable command-line driven graphing utility for Linux, Operating System (OS)/2, Microsoft (MS) Windows, Mac OS 10 (X), and VMS platforms. The source code is copyrighted but freely distributed. It currently supports batch mode operations. The home website can be found at http://www.gnuplot.info/.

This plot file, when used with the GNUPLOT program, is set up to generate a viewable plot file in Portable Document Format (PDF) format named "STARTING_CONDITIONS.pdf." It will contain the starting radial pressure, temperature, density, and Mach number. Additionally, at each point the solution generated from the downward running characteristic is also shown. The user can use this information to help determine if the initial solution is valid.

2.3.9 WALL_DEFINITION.dem File

This plot file is in the GNU plot format. This plot file, when used with the GNUPLOT program, is set up to generate a viewable plot file in PDF format named "WALL_DEFINITION.pdf." It will contain the final wall shape for both the inviscid and viscous solution. This file is a subset of the X_Y_FUN.dem file.

2.3.10 X_Y_FUN.dem File

This plot file is in the GNU plot format and is the main file used for viewing the results of the MOC code. This plot file, when used with the GNUPLOT program, is set up to generate up to eight different plots. These plots are in PDF format. The file is named "X_Y_FUN.pdf." The first plot displays the matching locations for each section of the nozzle. Each point is highlighted by a symbol. The second plot is identical to the first plot but without the symbol markers. This makes the plot easier to understand if the input

resolution is set very high. The next plot file is a colored image of Mach number flow within the nozzle. The fourth file is of axial velocity and the fifth is of radial velocity. The sixth plot is of flow angle. Strong waves with the flowfield are apparent in this plot. The seventh plot is of the wall shape. The four curves are of the original and smooth profiles of booth the viscous and inviscid solutions. The last plot is the slope of the original wall and the slope of the smoothed wall. This plot is useful at determining where the waves within the nozzle are hitting the wall.

2.3.11 fort.7, fort.8, fort.9, fort.10, fort.11, and fort.13 Files

These files contain the data needed to generate the plot files and are not intended to be used by the user. Once the users generate the PDF files, the "fort. *" files can be safely removed.

3. NAMELIST INPUTS

3.1 Overview

The main MOC input file contains user specified values arranged in a Fortran namelist. A namelist is a method of ordering input data into a routine. The name of the namelist is preceded by the "&" symbol, which is located in the second column of the input file. The namelist is terminated by either a "/" or "&END," also beginning in the second column of the input file. Comments may be placed within the file at any position but must be preceded with the "!" symbol.

There is only one namelist group. This group will define the options for different features, including code operation, flowfield and boundary conditions, and output. All supported variables in the namelist will be described along with any default information.

3.2 &INPUT Namelist Definition

Namelist &INPUT is used to define run time parameters and reference quantities. It is read only once from the "NOZZLE.inp" file. It contains several parameters. To simplify the definition of parameters, they have been grouped and are listed in Table 2 with a page reference for more detailed information.

Table 2. Namelist Input Summary

Tolerance	Page No.	Control Values	Page No.	Gas Properties	Page No.	Nozzle Properties	Page No.	Nozzle Solution	Page No.
ICOR	7	DELTA	8	PS	9	YT	10	IVL	11
E1	7	NI	8	TS	10	RTU	11	EMD	11
E2	7	NT	8	PA	10	RTD	11	I_FINAL	11
E3	7	KWRITE	8	G	10	AA	11	BL_SCALE	11
E4	8	IUNITS	8	RG	10	AE	11		
E5	8	ST	9	GL	10	XE	11		
		SMOOTH	9	GC	10				
		IORDER	9						

3.2.1 Tolerance Controls

The variables in this section deal with the tolerance controls within the code. In general, these values do not need to be changed.

&INPUT: ICOR Date Type: Integer

Description: Specify the number of Predictor/Corrector steps.

Value	Description	Default
> 0	Number of steps	30

&INPUT: E1 Date Type: Real

Description: Convergence tolerance for location, m.

Value	Description	Default
> = 0	Minimum error in spatial step error absolute	1.0E-06

- A value of zero means that this term will not be used to determine the convergence of the solution.

&INPUT: E2 Date Type: Real

Description: Fractional convergence tolerance for pressure.

Value	Description	Default
> = 0	Minimum fractional error in pressure	1.0E-06

- A value of zero means that this term will not be used to determine the convergence of the solution.
- Error is determined by comparing the difference in the pressure/predicted pressure to E2*predicted pressure.

&INPUT: E3 Date Type: Real

Description: Fractional convergence tolerance for density.

Value	Description	Default
> = 0	Minimum fractional error in density	1.0E-06

- A value of zero means that this term will not be used to determine the convergence of the solution.
- Error is determined by comparing the difference in the density/predicted density to E3*predicted density.

&INPUT: E4 Date Type: Real

Description: Fractional convergence tolerance for velocity.

Value	Description	Default
> = 0	Minimum fractional error in velocity	1.0E-06

- A value of zero means that this term will not be used to determine the convergence of the solution.
- Error is determined by comparing the difference in the velocity/predicted velocity to E4*predicted velocity.

&INPUT: E5 Date Type: Real

Description: Fractional convergence tolerance for flow angle.

Value	Description	Default
> = 0	Minimum fractional error in flow angle	1.0E-06

- A value of zero means that this term will not be used to determine the convergence of the solution.
- Error is determined by comparing the difference in the flow angle/predicted flow angle to E5*predicted flow angle.

3.2.2 Control Values

The values in this section deal with the solutions resolution.

&INPUT: DELTA Date Type: Real

Description: Axi or Planar calculation.

Value	Description	Default
0.0	Planer solution	
1.0	Axisymmetric solution	*

&INPUT: NI Date Type: Integer

Description: Specify number of points on the inflow plane.

Value	Description	Default
> 0	Number of points	21

&INPUT: NT Date Type: Integer

Description: Specify number of points that define the circular arc wall.

Value	Description	Default
> 0	Number of points	15

&INPUT: KWRITE Date Type: Integer

Description: Turn on the output.

Value	Description	Default
1	Output control	1

- Both the output and the errors are directed to Unit 6.

&INPUT: IUNITS Date Type: Integer

Description: Specify input units.

Value	Description	Default
2	Input Units	2

- All units must be input as standard SI units (N/m, m/s, kg/m^3, and so forth).

&INPUT: ST Date Type: Integer

Description: Specify radial start line spacing from wall.

Value	Description	Default
0	Even spacing	*
1	Quadratic spacing	

- The user is encouraged to use quadratic spacing as this puts the greatest number of points in the region of greatest change.

&INPUT: SMOOTH Date Type: Logical

Description: Smooth the spacing of points after each marching step.

Value	Description	Default
.F./.T.	Smooth the points	.False.

- If true, the solution will be linearly smoothed after each matching step but only AFTER the completion of the initial value start line solution. This option will help prevent points from building up into a strong wave. After each smoothing step, the code will try to save mass by modifying the density.

&INPUT: IORDER Date Type: Integer

Description: Specify the order of the least-squared curve fit for the wall profile.

Value	Description	Default
> 0	Order number	15

3.2.3 Gas Properties

The variables in this section define the properties of the nozzle flow gas.

&INPUT: PS Date Type: Real

Description: Specify the stagnation gas pressure (N/m).

Default: 70.0E+05

&INPUT: PA Date Type: Real

Description: Specify the ambient backpressure pressure (N/m).

Default: 0.0E+00

&INPUT: TS Date Type: Real

Description: Specify the stagnation temperature (K).

Default: 3000.0

&INPUT: G Date Type: Real

Description: Ratio of specific heats (Gamma).

Default: 1.4

&INPUT: RG Date Type: Real

Description: Universal gas constant (J/Kg-K).

Default: 287.04

&INPUT: GL Date Type: Real

Description: Unit conversion (M^2/M^2).

Default: 1.0

&INPUT: GC Date Type: Real

Description: Unit Conversion (M-Kg/N-s^2).

Default: 1.0

3.2.4 Nozzle Wall Properties

The variables in this section deal with the initial shape of the nozzle wall.

&INPUT: YT Date Type: Real

Description: Nozzle throat radius (m).

Default: 1.0

&INPUT: RTU Date Type: Real

Description: Radius of curve upstream of throat (m).

Default: 2.0

&INPUT: RTD Date Type: Real

Description: Radius of curve downstream of throat (m).

Default: 0.5

&INPUT: AA Date Type: Real

Description: Attachment angle (degree) of the last point on the circular arc wall.

Default: 10.0

&INPUT: AE Date Type: Real

Description: Exit angle of nozzle assuming no predicted wall.

Default: 10.0

&INPUT: XE Date Type: Real

Description: Stopping location of nozzle wall.

Default: 10.0

3.2.5 Nozzle Solution Parameters

The variables in this section control additional parameters that affect the solution.

&INPUT: EMD Date Type: Real

Description: Exit design Mach number of the nozzle.

Default: 5.0

&INPUT: BL_SCALE Date Type: Real

Description: Scale factor for the boundary layer correction wall shape. For nozzles with large exit design Mach numbers, this factor may need to be increased to 2.1.

Default: 1.0

&INPUT: I_FINAL Date Type: Integer

Description: Number of points on final characteristic line.

Value	Description	Default
> 0	Number of points	50

&INPUT: NOZ Date Type: Integer

Description: Nozzle shape.

Value	Description	Default
0	Final nozzle shape will be determined by the MOC code.	*
1	Nozzle shape will be limited to circular arc and second order quadratic wall only	

&INPUT: IVL Date Type: Integer

Description: Start line input.

Value	Description	Default
0	Transonic start line	*
1	User inputted start line	

- If NOZ = 1, user must supply a "PROFILE.dat" file. This file must start with conditions on the centerline and move up to the wall. It must be in the format of I, J, K, X, Y, Z, Rho, U, V, W, and P; where I, J, and K are integer index points; X, Y, and Z are the point locations (Z is not used); Rho is the density; U, V, and W are the velocity components; and P is the pressure. The temperature is calculated by the equation of state.

REFERENCES

1. Zucrow, M. J. and Hoffman, J. D., Gas Dynamics, Volume I, John Wiley & Sons, Inc., New York, 1976.

2. Zucrow, M. J. and Hoffman, J. D., Gas Dynamics, Volume II, John Wiley & Sons, Inc., New York, 1977.

3. Adams, J.C.; Brainerd, W.S.; Martin, J.Y.; Smith, B.T.; and Wagener, J.L., FORTRAN 95 Handbook, The MIT Press, Cambridge, Massachusetts, 1997.

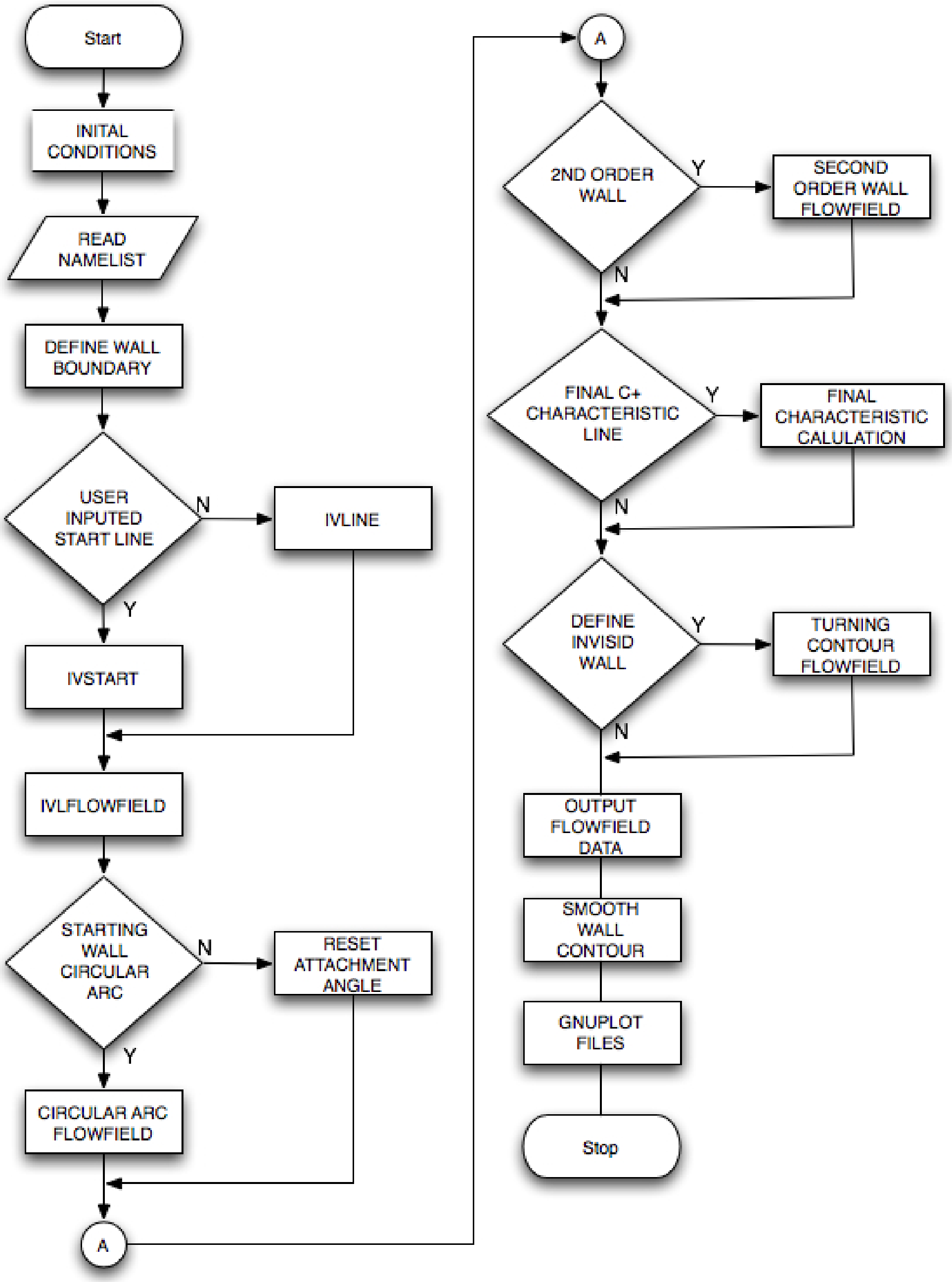
4. GNUPLOT Homepage, January 2012, <http://www.gnuplot.info>

LIST OF ACRONYMS AND ABBREVIATIONS

2-D	Two Dimensional
ASCII	American Standard Code for Information Interchange
BL	Boundary Layer
MOC	Method of Characteristic
MS	Microsoft
OS	Operating System
PDF	Portable Document Format
VMS	Virtual Memory System
X	Ten

APPENDIX A
FLOW CHART

The flow chart is offered to show the top-level processes of the Method of Characteristic (MOC) code. The details of these processes are given in the FORTRAN coding of Appendix B.

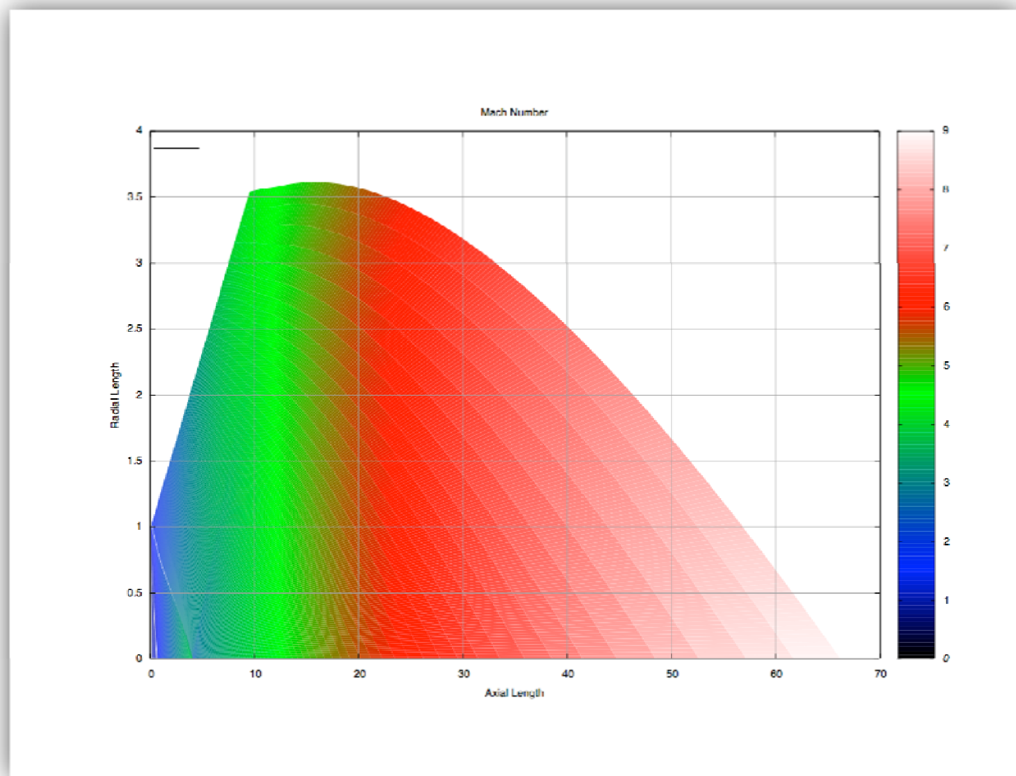


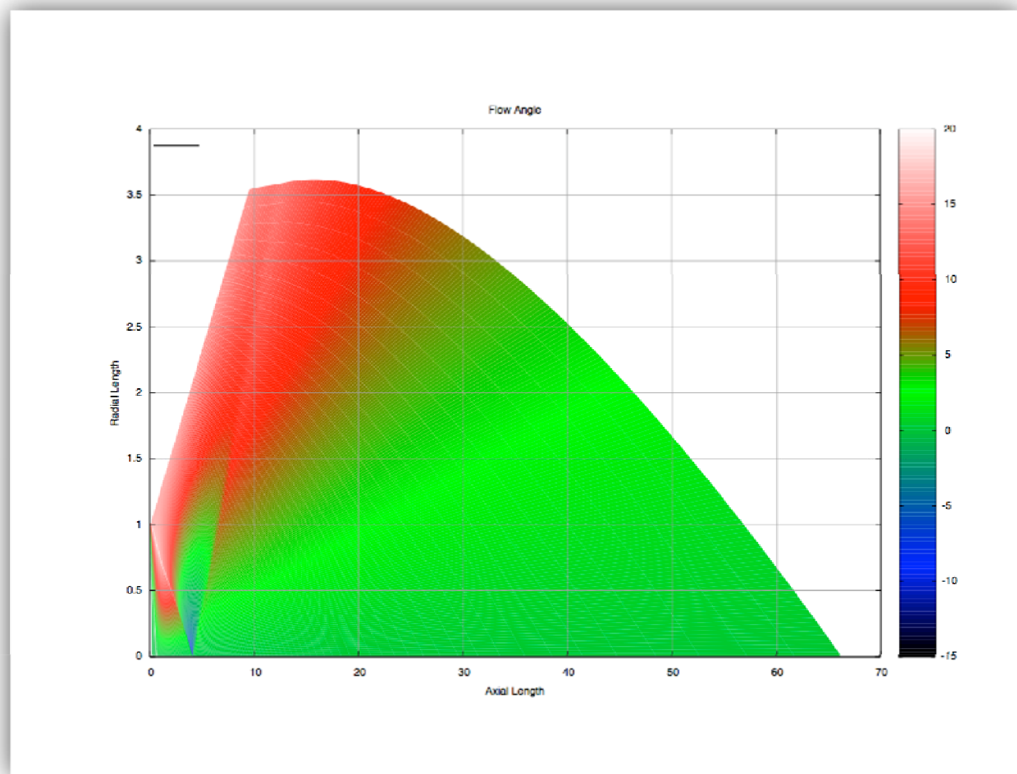
APPENDIX B
EXAMPLE 1

Example 1 is set up to model an axisymmetric nozzle. The code will calculate a transonic start line based on input conditions. The nozzle wall is defined at all locations with a circular arc and a second-order curve. The flowfield calculation is stopped before the design Mach number is reached. Data smoothing is turned on. This case is designed to emphasize the presence of the development of strong waves near the centerline. These waves result in grid lines crossing at about 5-meters downstream. The user should take note of the variables highlighted in red. Plots of the Mach number and flow angle are included.

```
 &INPUT
!...
!... Set Tolerances
!...
      ICOR   = 30        !predictor corrector term
      E1     = 1.0E-06   !m  X space error
      E2     = 1.0E-06   !   Pressure error
      E3     = 1.0E-06   !   Density error
      E4     = 1.0E-06   !   Velocity error
      E5     = 1.0E-06   !   flow angle error
!...
!... Set Control values
!...
      DELTA  = 1.0        !1 axi, 0 planer (Mass flux not working correctly)
      NI     = 81         !NUMBER OF RADIAL POINTS ON INFLOW PLANE (Max 99)
      NT     = 31         !NUMBER OF CIRULAR ARC POINTS
      KWRITE = 1
      IUNITS = 2
      SMOOTH = .TRUE.     !EVENLY SPACE OUT THE DATA AFTER EACH SECTION
      ST     = 1.0        !0.0 Even space start line, 1.0 quadratic spaced
      IORDER = 15
!...
!... Set Gas Properties
!...
      PS     = 1600.000E+06 !gas pressure
      TS     = 1.000000E+03 !gas temperature
      PA     = 0.0          !ambient pressure
      G      = 1.4          !gamma
      RG     = 287.040      !J/KG-K
      GC     = 1.0          !M-KG/N-S^2
      GL     = 1.0          !M^2/M^2
!...
!... Set Nozzle Properties
!...
      YT     = 1.000 !nozzle radius
      RTU    = 2.000 !radius of curve upstream of throat
      RTD    = 0.500 !radius of curve downstream of throat
      AA     = 15.0  !attachment angle
      AE     = 15.0  !exit angle
      XE     = 10.0  !exit location
!...
!... Set Nozzle Solution
!...
     IVL      = 0     !0 transonic start, 1 User defined input
     EMD      = 5.0   !design mach number
     NOZ      = 1     !1 nozzle shape inputted, 0 nozzle shape from MOC
     I_FINAL  = 201   !number of points on final chara line
     BL_SCALE = 2.1   !boundary layer scalar

 /
```

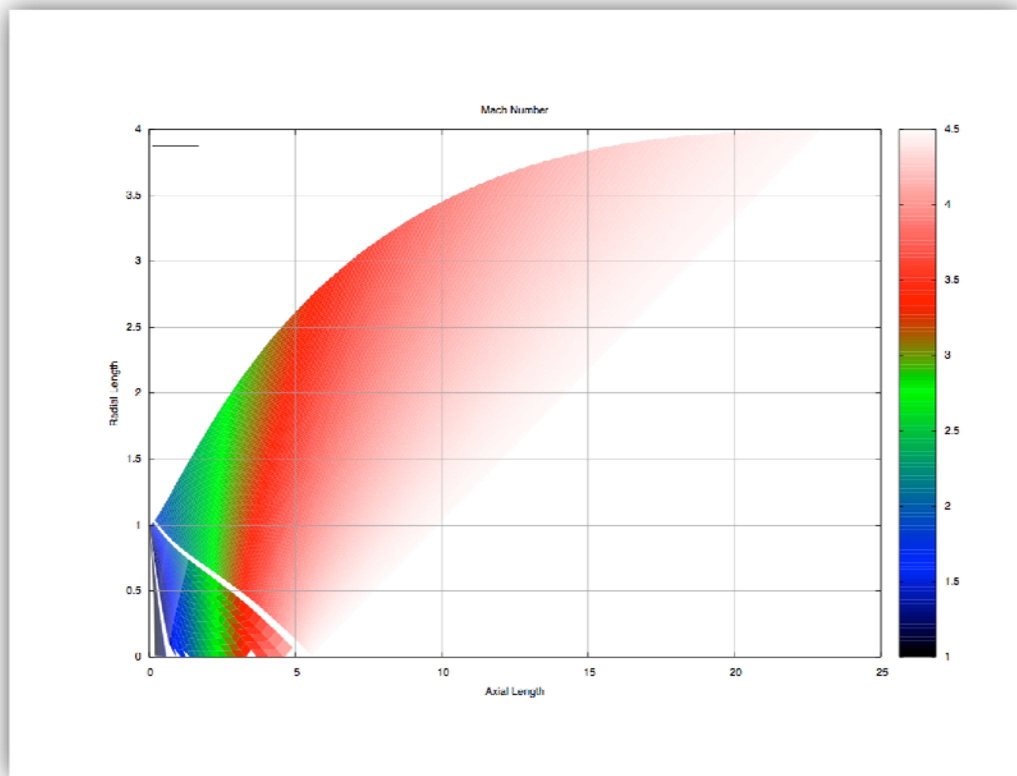

Mach Number
Radial Length
Axial Length


Flow Angle
Radial Length
Axial Length

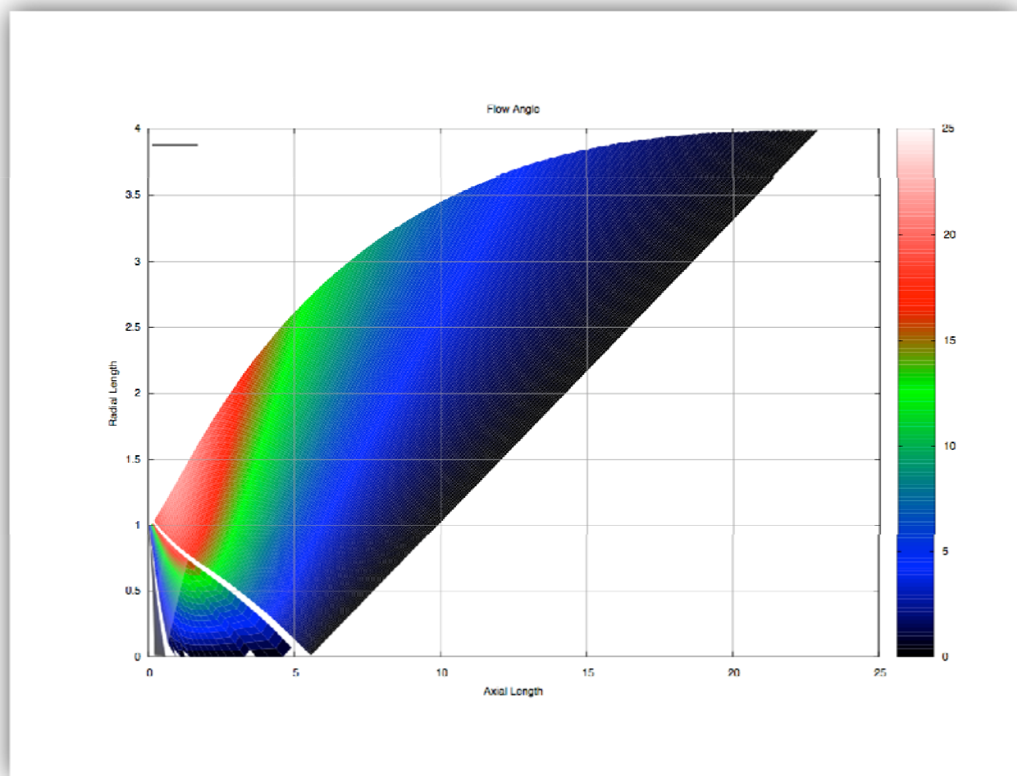
APPENDIX C
EXAMPLE 2

Example 2 is set up to model an axisymmetric nozzle. The code will calculate a transonic start line based on input conditions. The nozzle wall is initially defined with a circular arc. The flowfield calculation is stopped when the design Mach number on the centerline is reached. There is no data smoothing. This case is designed to emphasize the large attachment angle and the creation of the inviscid wall. The user should take note of the variables highlighted in red. Plots of the Mach number and flow angle are included.

```
 &INPUT
!...
!... Set Tolerances
!...
      ICOR   = 30        !predictor corrector term
      E1     = 1.0E-06  !m  X space error
      E2     = 1.0E-06  !   Pressure error
      E3     = 1.0E-06  !   Density error
      E4     = 1.0E-06  !   Velocity error
      E5     = 1.0E-06  !   flow angle error
!...
!... Set Control values
!...
      DELTA  = 1.0       !1 axi, 0 planer
      NI     = 71        !NUMBER OF RADIAL POINTS ON INFLOW PLANE (Max 99)
      NT     = 35        !NUMBER OF CIRULAR ARC POINTS
      KWRITE = 1
      IUNITS = 2
      SMOOTH = .FALSE.   !EVENLY SPACE OUT THE DATA AFTER EACH SECTION
      ST     = 1.0       !0.0 Even space start line, 1.0 quadratic spaced
      IORDER = 30
!...
!... Set Gas Properties
!...
      PS     = 1600.000E+06 !gas pressure
      TS     = 1.000000E+03 !gas temperature
      PA     = 0.0          !ambient pressure
      G      = 1.4          !gamma
      RG     = 287.040      !J/KG-K
      GC     = 1.0          !M-KG/N-S^2
      GL     = 1.0          !M^2/M^2
!...
!... Set Nozzle Properties
!...
      YT     = 1.000 !nozzle radius
      RTU    = 2.000 !radius of curve upstream of throat
      RTD    = 0.500 !radius of curve downstream of throat
      AA     = 35.0  !attachment angle
      AE     = 15.0  !exit angle
      XE     = 10.0  !exit location
!...
!... Set Nozzle Solution
!...
     IVL      = 0     !0 transonic start, 1 User defined input
     EMD      = 4.5   !design mach number
     NOZ      = 0     !0 calculate a MOC wall, 1 no MOC wall calculated
     I_FINAL  = 351   !number of points on final chara line
     BL_SCALE = 2.1   !boundary layer scalar
 /
```



Mach Number
Radial Length
Axial Length

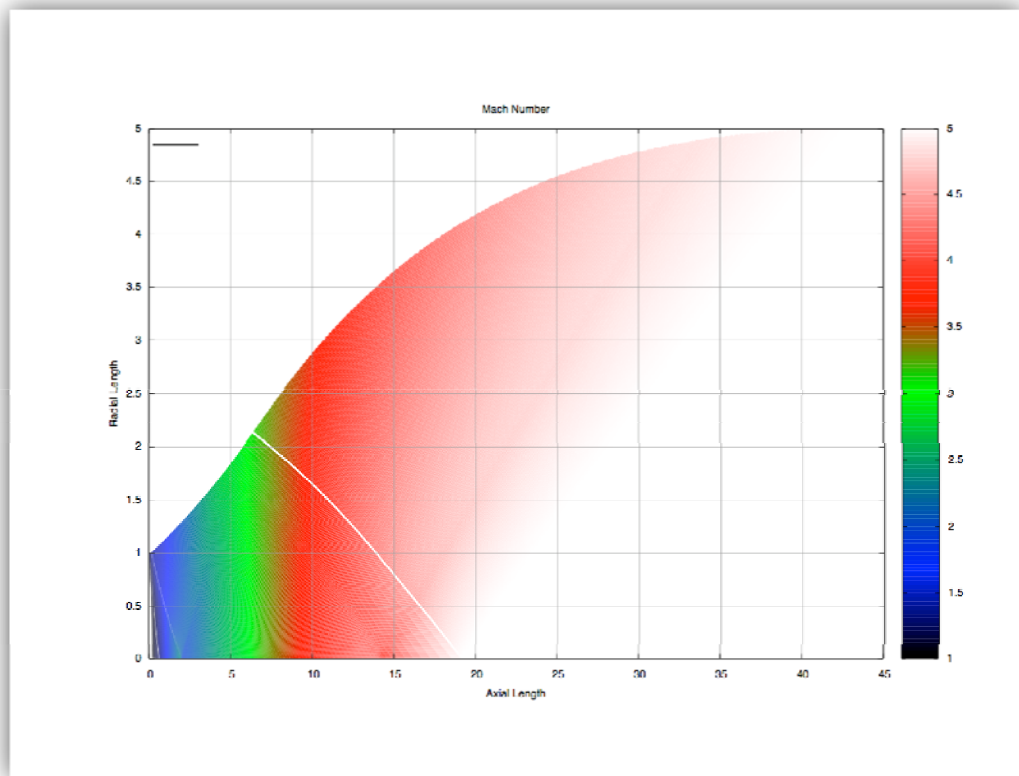


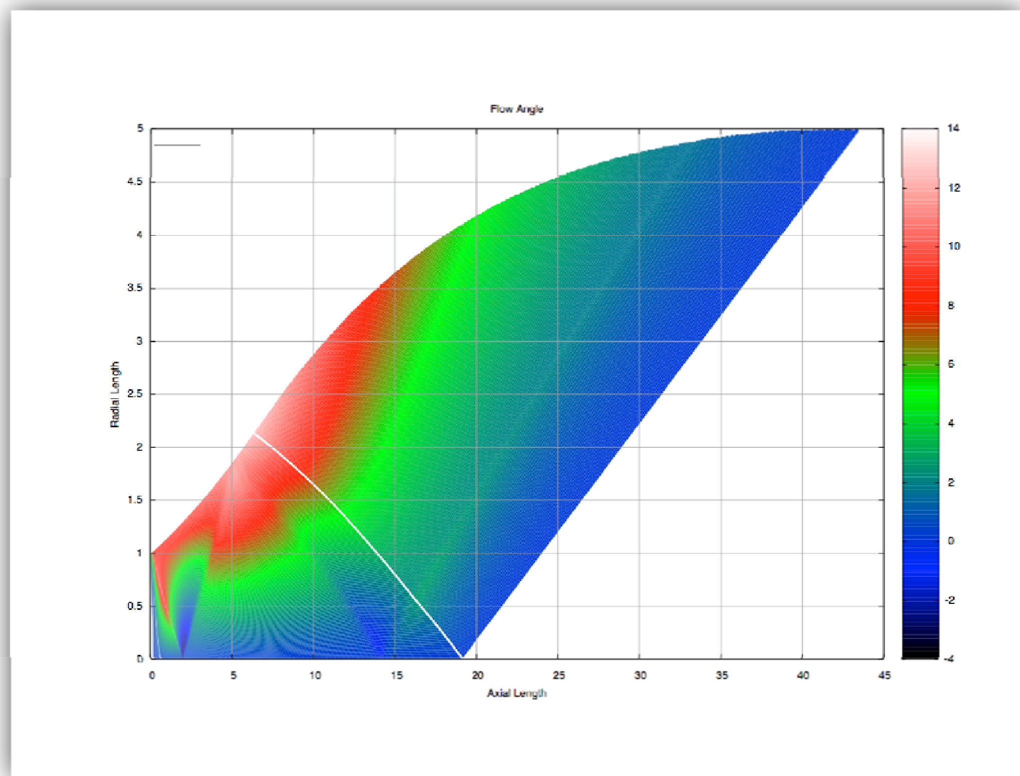
Flow Angle
Radial Length
Axial Length

APPENDIX D
EXAMPLE 3

Example 3 is set up to model an axisymmetric nozzle. The code will calculate a transonic start line based on input conditions. The nozzle wall is initially defined with a circular arc, then a second-order quadratic, and then a calculated inviscid wall. The flowfield calculation is stopped when the design Mach number on the centerline is reached. Data smoothing is on. This case is designed to emphasize how the inviscid wall cancels out the strong wave in the flowfield. The user should take note of the variables highlighted in red. Plots of the Mach number and flow angle are included.

```
&INPUT
!...
!... Set Tolerances
!...
      ICOR   = 30        !predictor corrector term
      E1     = 1.0E-06   !m  X space error
      E2     = 1.0E-06   !   Pressure error
      E3     = 1.0E-06   !   Density error
      E4     = 1.0E-06   !   Velocity error
      E5     = 1.0E-06   !   flow angle error
!...
!... Set Control values
!...
      DELTA  = 1.0        !1 axi, 0 planer
      NI     = 71         !NUMBER OF RADIAL POINTS ON INFLOW PLANE (Max 99)
      NT     = 31         !NUMBER OF CIRULAR ARC POINTS
      KWRITE = 1
      IUNITS = 2
      ST     = 1.0        !0.0 Even space start line, 1.0 quadratic spaced
      SMOOTH = .TRUE.     !EVENLY SPACE OUT THE DATA AFTER EACH SECTION
!...
!... Set Gas Properties
!...
      PS     = 1600.000E+06 !gas pressure
      TS     = 1.000000E+03 !gas temperature
      PA     = 0.0       !ambient pressure
      G      = 1.4       !gamma
      RG     = 287.040   !J/KG-K
      GC     = 1.0       !M-KG/N-S^2
      GL     = 1.0       !M^2/M^2
!...
!... Set Nozzle Properties
!...
      YT     = 1.000 !nozzle radius
      RTU    = 2.000 !radius of curve upstream of throat
      RTD    = 1.000 !radius of curve downstream of throat
      AA     = 8.0   !attachment angle
      AE     = 15.0  !exit angle
      XE     = 10.0  !exit location
!...
!... Set Nozzle Solution
!...
     IVL      = 0     !0 transonic start, 1 User defined input
     EMD      = 5.0   !design mach number
     NOZ      = 0     !1 nozzle shape inputted, 0 nozzle shape from MOC
     I_FINAL  = 301   !number of points on final chara line
     BL_SCALE = 2.1   !boundary layer scalar
     IORDER   = 45
 /
```

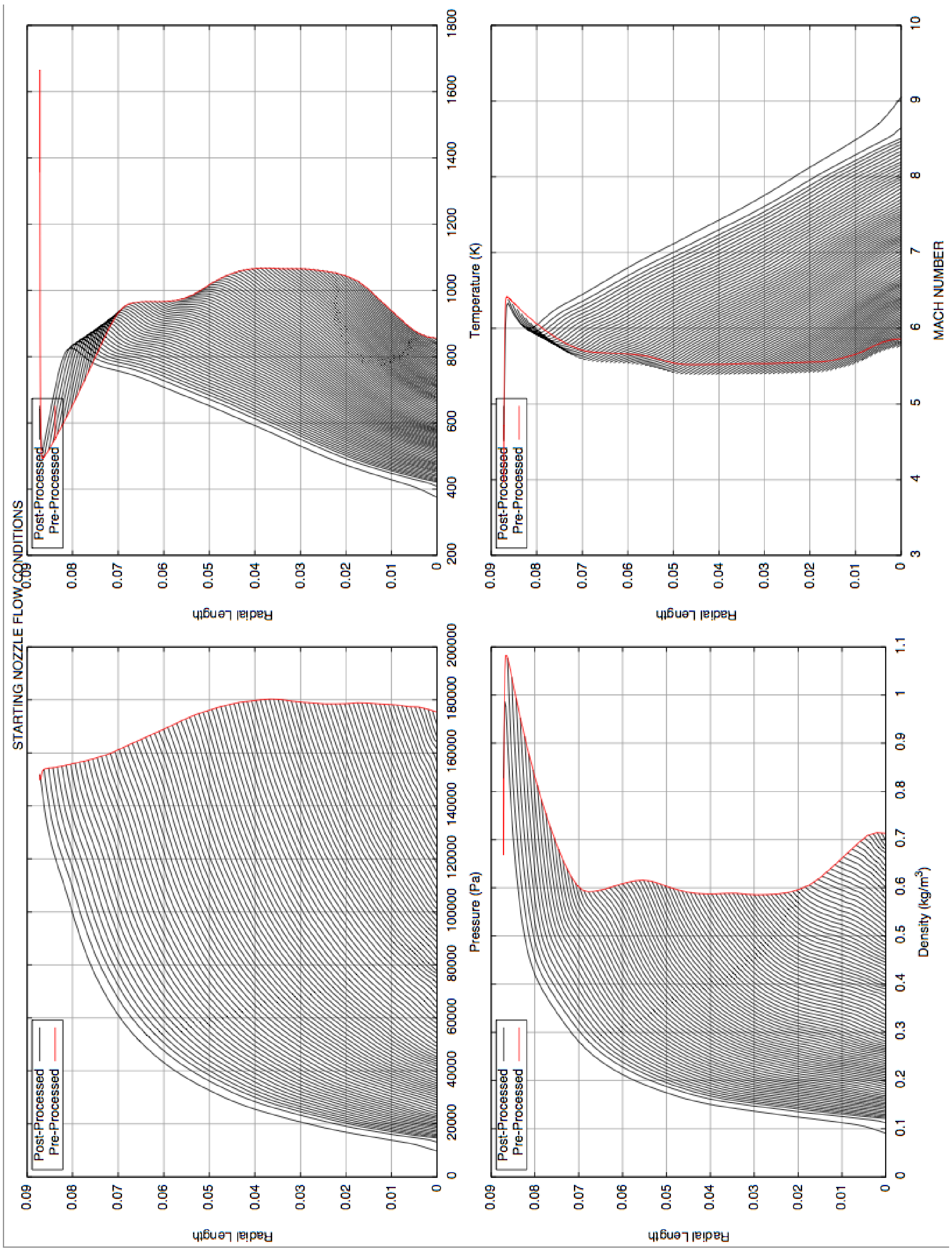

Mach Number
Radial Length
Axial Length

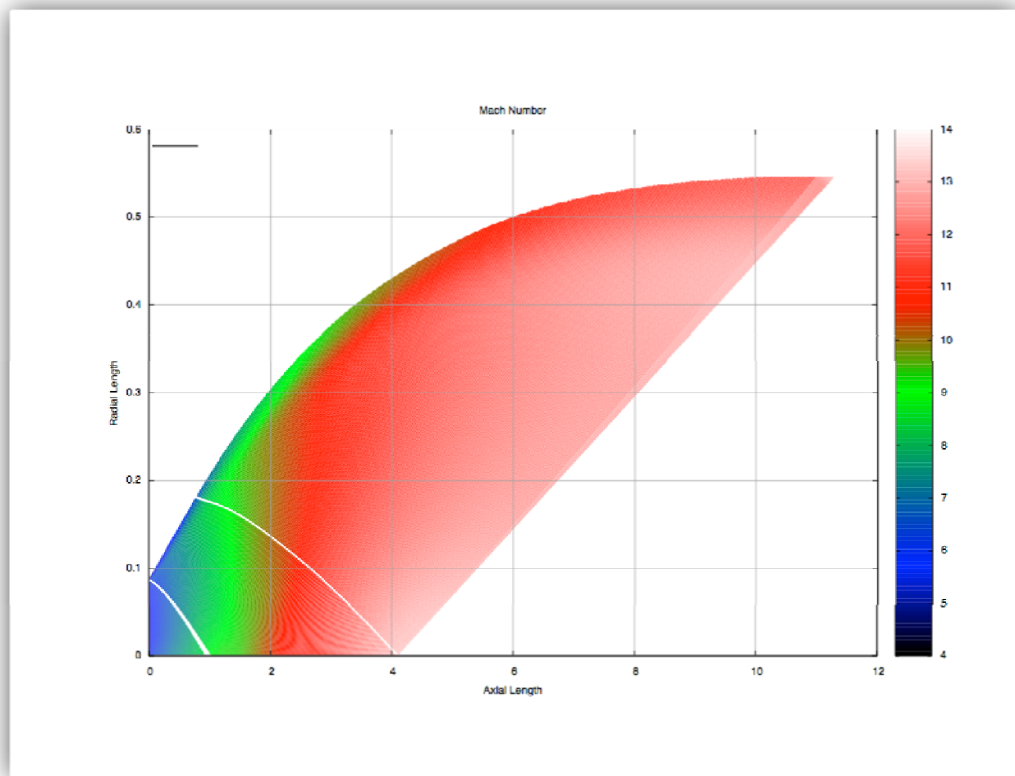

Flow Angle
Radial Length
Axial Length

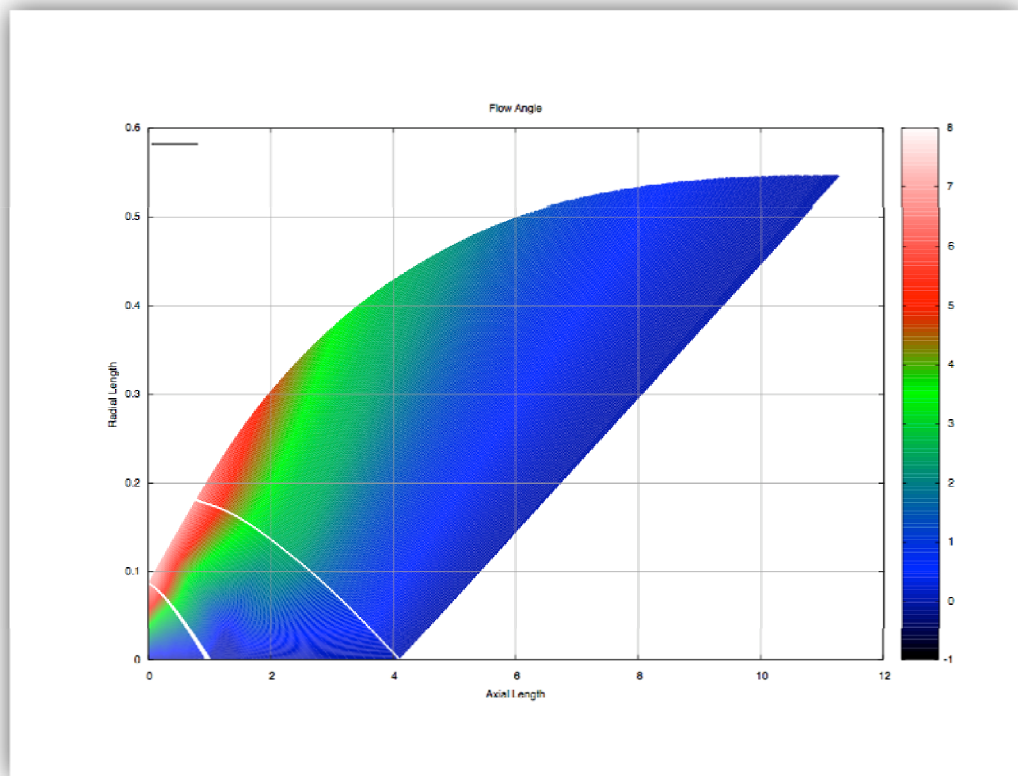
APPENDIX E
EXAMPLE 4

Example 4 is set up to model an axisymmetric nozzle. The user will specify the start line input conditions. The nozzle wall is initially defined with a circular arc, then a second-order quadratic, and then a calculated inviscid wall. The flowfield calculation is stopped when the design Mach number on the centerline is reached. Data smoothing is on. This case is designed to show that the Method of Characteristic (MOC) code can start with an arbitrary start line. The exit solution near the wall does not match the design Mach number. This is due to a cold profile that is near the wall. The user should take note of the variables highlighted in red. Plots of the start line (including its initial flowfield solution), Mach number, and flow angle are included.

```
&INPUT
!...
!... Set Tolerances
!...
      ICOR   = 30         !predictor corrector term
      E1     = 1.0E-06   !m  X space error
      E2     = 1.0E-06   !   Pressure error
      E3     = 1.0E-06   !   Density error
      E4     = 1.0E-06   !   Velocity error
      E5     = 1.0E-06   !   flow angle error
!...
!... Set Control values
!...
      DELTA  = 1.0        !1 axi, 0 planer
      NI     = 91         !NUMBER OF RADIAL POINTS ON INFLOW PLANE (Max 99)
      NT     = 15         !NUMBER OF CIRULAR ARC POINTS
      KWRITE = 1
      IUNITS = 2
      SMOOTH = .TRUE.     !EVENLY SPACE OUT THE DATA AFTER EACH SECTION
      ST     = 1.0        !0.0 Even space start line, 1.0 quadratic spaced
      IORDER = 25
!...
!... Set Gas Properties
!...
      PS     = 1600.000E+06 !gas pressure chamber
      TS     = 1.000000E+03 !gas temperature chamber
      PA     = 0.0          !ambient pressure
      G      = 1.4          !gamma
      RG     = 287.040      !Gas constant J/KG-K
      GC     = 1.0          !M-KG/N-S^2
      GL     = 1.0          !M^2/M^2
!...
!... Set Nozzle Properties
!...
      YT     = 1.000 !nozzle radius
      RTU    = 2.000 !radius of curvature upstream of throat
      RTD    = 0.500 !radius of curvature downstream of throat
      AA     = 15.0  !attachment angle - angle at which flow transition to MOC wall
      AE     = 6.259 !exit angle
      XE     = 2.000 !exit location
!...
!... Set Nozzle Solution
!...
     IVL       = 1       !0 transonic start, 1 User defined input from "Profile.dat"
     EMD       = 13.25   !design exit mach number
     NOZ       = 0       !1 nozzle shape inputted, 0 nozzle shape from MOC
     I_FINAL   = 351     !number of points on final chara line
     BL_SCALE = 2.1      !boundary layer scalar
 /
```


STARTING NOZZLE FLOW CONDITIONS
Post-Processed
Pre-Processed
Radial Length
Pressure (Pa)
Temperature (K)
Density (kg/m^3)
MACH NUMBER

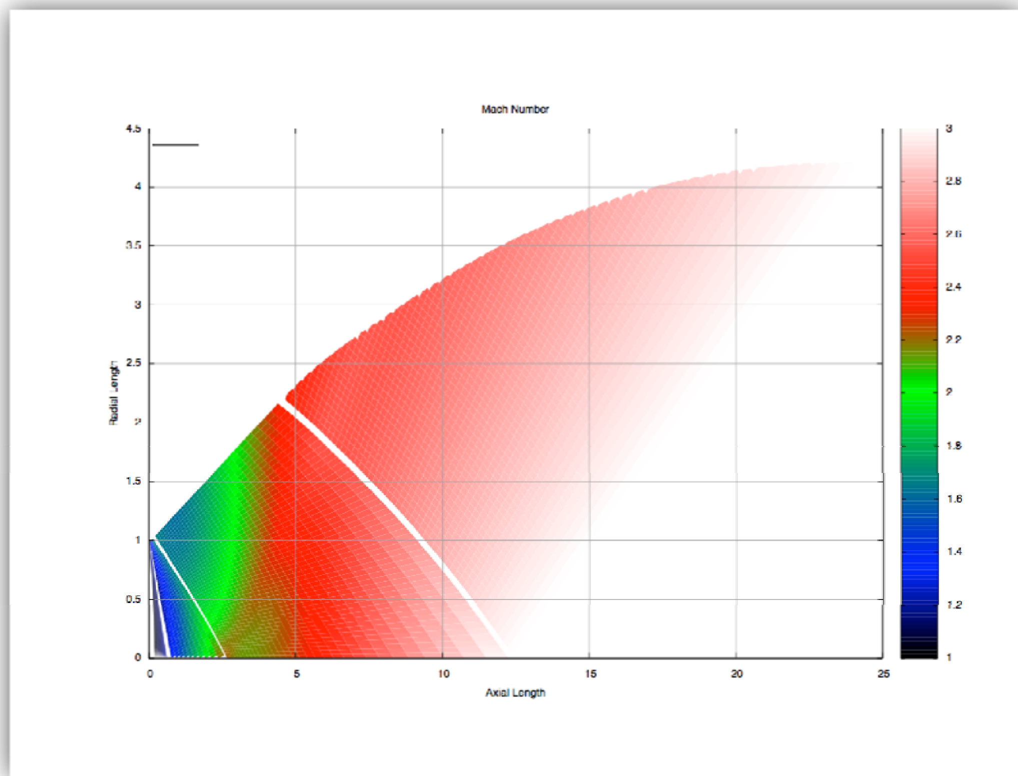

Mach Number
Radial Length
Axial Length
0
0.1
0.2
0.3
0.4
0.5
0.6
2
4
6
8
10
12
14
13
12
11
10
9
8
7
6
5
4

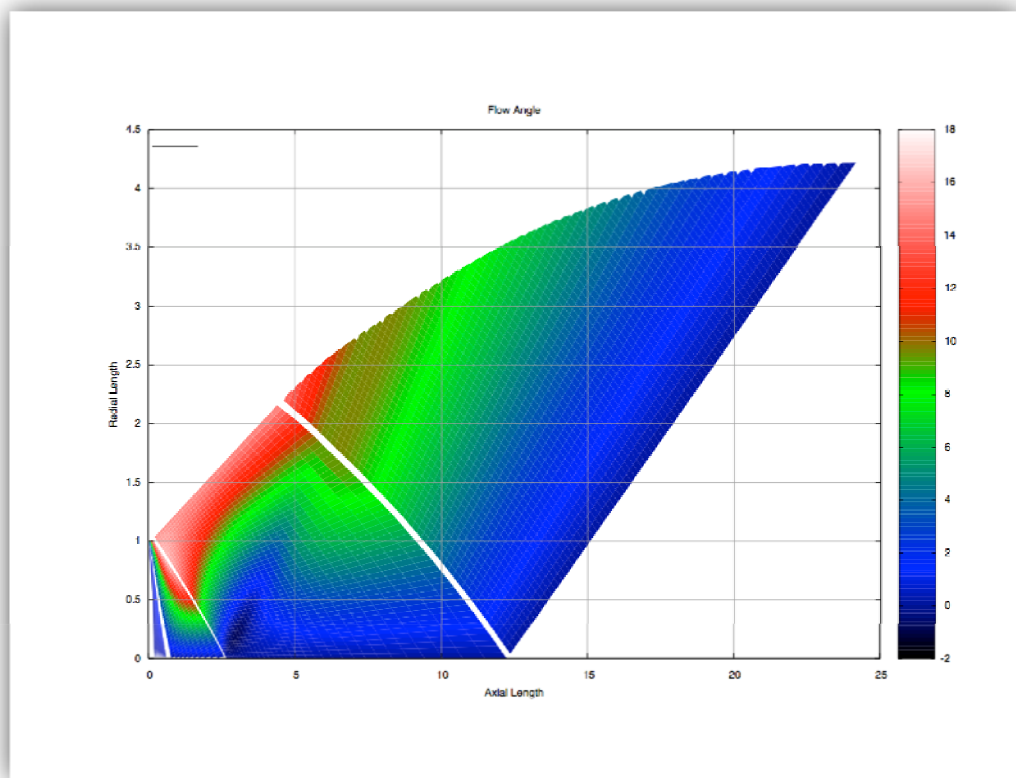

Flow Angle
Radial Length
Axial Length
0
0.1
0.2
0.3
0.4
0.5
0.6
2
4
6
8
10
12
8
7
6
5
4
3
2
1
0
-1

APPENDIX F
EXAMPLE 5

Example 5 is set up to model a planar nozzle. The code will calculate a transonic start line based on input conditions. The nozzle wall is initially defined with a circular arc, then a second-order quadratic, and then a calculated inviscid wall. The flowfield calculation is stopped when the design Mach number on the centerline is reached. Data smoothing is on. This case is designed to emphasize how the inviscid wall cancels out the strong wave in the flowfield and to show that planar nozzles have large exit radii. The user should take note of the variables highlighted in red. Plots of the Mach number and flow angle are included.

```
&INPUT
!...
!... Set Tolerances
!...
      ICOR   = 30        !predictor corrector term
      E1     = 1.0E-06   !m  X space error
      E2     = 1.0E-06   !   Pressure error
      E3     = 1.0E-06   !   Density error
      E4     = 1.0E-06   !   Velocity error
      E5     = 1.0E-06   !   flow angle error
!...
!... Set Control values
!...
      DELTA  = 0.0       !1 axi, 0 planer (Mass flux not working correctly)
      NI     = 21        !NUMBER OF RADIAL POINTS ON INFLOW PLANE (Max 99)
      NT     = 11        !NUMBER OF CIRULAR ARC POINTS (Should = AA)
      KWRITE = 1
      IUNITS = 2
      SMOOTH = .TRUE.    !EVENLY SPACE OUT THE DATA AFTER EACH SECTION
      ST     = 1.0       !0.0 Even space start line, 1.0 quadratic spaced
      IORDER = 35
!...
!... Set Gas Properties
!...
      PS     = 1600.000E+06 !gas pressure
      TS     = 1.000000E+03 !gas temperature
      PA     = 0.0          !ambient pressure
      G      = 1.4          !gamma
      RG     = 287.040      !J/KG-K
      GC     = 1.0          !M-KG/N-S^2
      GL     = 1.0          !M^2/M^2
!...
!... Set Nozzle Properties
!...
      YT     = 1.000 !nozzle radius
      RTU    = 2.000 !radius of curve upstream of throat
      RTD    = 0.500 !radius of curve downstream of throat
      AA     = 15.0  !attachment angle
      AE     = 15.0  !exit angle
      XE     = 20.0  !exit location
!...
!... Set Nozzle Solution
!...
     IVL      = 0     !0 transonic start, 1 User defined input
     EMD      = 3.0   !design mach number
     NOZ      = 0     !1 nozzle shape inputted, 0 nozzle shape from MOC
     I_FINAL  = 101   !number of points on final chara line
     BL_SCALE = 2.1   !boundary layer scalar
 /
```


Mach Number
Radial Length
Axial Length


Flow Angle
Radial Length
Axial Length

APPENDIX G
ERROR MESSAGES

1. Error Code: "STOPPING CODE: RESET NII TO XXX"

   The variable NII in file "MOC.PAR" needs to be reset to XXX. This error normal occurs if the variable 'I_FINAL' in name list $INPUT is set too large. (Source code: FINAL_CHAR_LINE.f and NOZZLE.f)

2. Error Code: "Cannot find namelist &INPUT in file "NOZZLE.inp""

   The namelist &INPUT was not found in the input file. Make sure that the "&" starts in the second column. (Source code: READ_NAMLIST.f)

3. Error Code: "THE RATIO OF RTU/YT IS INVALID FOR SAUERS METHOD!"

   The user input values of RTU and YT are invalid for the code to predict a start line profile. The user must increase the ratio RTU/YT to be greater than 1.0. Ideally this ratio should be greater than 2.0. For values greater than 1 but less than 2, the code will print out a warning message that the ratio is only marginally acceptable. This is a limitation of the use of Sauers method[1,2] for the prediction of the start line profile. (Source code: IVLINE.f)

4. Error Code: "File Profile.dat does not exist."

   The user has set the variable "IVL" to 1 in the namelist there by requesting an inputted start line profile. The file that contains the start line, "Profile.dat," could not be found within the working directory. (Source code: IVSTART.f)

5. Error Code: "Stopping the code: Variable NII needs to be increased."

   The variable NII in file "MOC.PAR" needs to be increased to a larger value. (Source code: SECOND_ORDER_WALL_FLOWIELD.f)

6. Error Code: "ERROR: CAN NOT DECODE FLOW ANGLE!"

   This error is common when the input resolution is not sufficient to resolve the flowfield. To fix this, the user can increase the variable "NI" in the input file. If this error still remains, the user can turn on flow smoothing by setting the logical variable "SMOOTH" to TRUE. (Source code: INTERIOR_POINT.f and TURNING.f)

7. Error Code: "Stopping the code: Reached the ICOR user set limit!"

   This error occurs if the variable "ICOR" is set too low or one of the error terms "E1" through "E5," is set too high for the local flow conditions. In general, the variable "ICOR" should be set to 30 and the individual error terms should be lowered. (Source code: TURNING.f)

APPENDIX H
FORTRAN 90 CODE

```
      PROGRAM MOC
C
C*********************************************************************
C*                                                                   *
C*     TERMINOLOGY FOR SUPERSONIC FLOW METHOD OF CHARACTERISTICS     *
C*                                                                   *
C* CONTROL VARIABLE:                                                 *
C* ----------------                                                  *
C*                                                                   *
C* DELTA  = '0' FOR PLANER FLOW                                      *
C*        = '1' AXISYMMETRIC FLOW                                    *
C* ICOR   = NUMBER OF APPLICATIONS OF THE CORRECTOR DESIRED          *
C* E1     = CONVERGENCE TOLERANCE FOR LOCATION, M (IN)               *
C* E2     = CONVERGENCE TOLERANCE FOR VELOCITY, M/S (FT/SEC)         *
C* GC     = 1.0 M-KG/N-S^2 OR 32.174 FT-LBM/LBF-S^2                  *
C* GL     = 1.0 M^2/M^2 OR 144.0 IN^2/FT^2                           *
C* ST     = 0.0 EVEN SPACE STARTLINE, 1.0 QUADRATIC SPACED           *
C*                                                                   *
C* GAS THERMODYNAMIC PROPERTIES & STAGNATION PROPERTIES:             *
C* ----------------------------------------------------              *
C*                                                                   *
C* G      = RATIO OF SPECIFIC HEATS                                  *
C* RG     = GAS CONSTANT, J/KG-K (FT-LBF/LBM-R)                      *
C* TS     = STAGNATION TEMPERATURE, K (R)                            *
C* PS     = STAGNATION PRESSURE, N/M^2 (LBF/IN^2)                    *
C* PA     = AMBIENT PRESSURE, N/M^2 (LBF/IN^2)                       *
C*                                                                   *
C* FLOW FIELD PROPERTIES:                                            *
C* ---------------------                                             *
C*                                                                   *
C* X      = AXIAL COORDINATE, M (IN)                                 *
C* Y      = RADIAL COORDINATE, M (IN)                                *
C* U      = AXIAL VELOCITY, M/S (FT/S)                               *
C* V      = RADIAL VELOCITY, M/S (FT/S)                              *
C* Q      = VELOCITY MAGNITUDE, M/W (FT/S)                           *
C* A      = FLOW ANGLE, RAD                                          *
C* P      = STATIC PRESSURE, N/M^2 (LBF/IN^2)                        *
C* R      = STATIC DENSITY, KG/M^3 (LBM/FT^3)                        *
C* T      = STATIC TEMPERATURE, K (R)                                *
C* C      = SPEED OF SOUND, M/S (FT/S)                               *
C* M      = MACH NUMBER                                              *
C* EMD    = DESIGN MACH NUMBER                                       *
C* 1,2,3, = DENOTES PROPERTIES AT POINTS                             *
C*                                                                   *
C* TERMINOLOGY EMPLOYED:                                             *
C* --------------------                                              *
C*                                                                   *
C* L      = TAN(THETA+-ALPHA)                                        *
C* Q      = (U^2-C^2), M^2/S^2 (FT^2/S^2)                            *
C* R      = 2UV-L(U^2-C^2) M^2/S^2 (FT^2/S^2)                        *
C* S      = DELTA*C^2*V/Y, M^2/S^3 (FT^2/SEC^3-IN)                   *
C* T      = S*DEL(X)+Q*U+R*V, M^3/S^3 (FT^3/S^3)                     *
C* +/-    = DENOTES + OR - CHARACTERISTIC CURVE                      *
C*                                                                   *
C*********************************************************************
C
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C
C********************************************************************
C**                                                                **
C**   Function Statements                                          **
C**                                                                **
C********************************************************************
C
      TM(B) = B / TZERO
      ZMUIFD(B) = ZMZERO*TM(B)**1.5*((1.+CONSU)/(TM(B)+CONSU))
      RE(B) = R(1) * Q(1) * X(1) / ZMUIFD(B)
      BL(B) = RE(B)**(1./8.)
C
C********************************************************************
C**                                                                **
C**   OUTPUT THE HEADER STATEMENTS                                 **
C**                                                                **
C********************************************************************
C
```

```
PRINT*,' '
      PRINT*,'C********************************************************'
      PRINT*,'C*                                                      *'
      PRINT*,'C*           ##     ##  #####   #####                   *'
      PRINT*,'C*           # #   # # #     # #     #                  *'
      PRINT*,'C*           #  # #  # #     # #                        *'
      PRINT*,'C*           #   #   # #     # #                        *'
      PRINT*,'C*           #       # #     # #                        *'
      PRINT*,'C*           #       # #     # #     #                  *'
      PRINT*,'C*           #       #  #####   #####                   *'
      PRINT*,'C*                                                      *'
      PRINT*,'C*           METHOD OF CHARACTERISTICS                  *'
      PRINT*,'C*                 (VERSION 1.0)                        *'
      PRINT*,'C*         MODIFICATION DATE: 24 FEBUARY 2012           *'
      PRINT*,'C*                                                      *'
      PRINT*,'C*            WRITTEN BY: K. D. KENNEDY                 *'
      PRINT*,'C*  AERODYNAMICS TECHNOLOGY BRANCH (RMMR-SSM-A)         *'
      PRINT*,'C*      US ARMY AVIATION & MISSILE COMMAND              *'
      PRINT*,'C*      REDSTONE ARSENAL, ALABAMA 35898-5252            *'
      PRINT*,'C*                                                      *'
      PRINT*,'C*  BASED ON: GAS DYNAMICS, VOL 2 MULTIDIMENSIONAL FLOW *'
      PRINT*,'C*        BY: M ZUCROW AND J HOFFMAN                    *'
      PRINT*,'C*                                                      *'
      PRINT*,'C********************************************************'
      PRINT*,' '

C
C***********************************************************************
C*                                                                     *
C*    OPEN THE INPUT AND OUTPUT FILES                                  *
C*                                                                     *
C***********************************************************************
C
      OPEN(UNIT = 2 , FILE = 'NOZZLE.inp', FORM='FORMATTED')
      OPEN(UNIT = 3 , FILE = 'WALL.dat',   FORM = 'FORMATTED')
      OPEN(UNIT = 4 , FILE = 'CENTERLINE.dat', FORM = 'FORMATTED')
      OPEN(UNIT =32 , FILE = 'WALL_BOUNDARY_CORRECTED.dat',  FORM = 'FORMATTED')
C
C***********************************************************************
C*                                                                     *
C*    READ IN THE NAMELIST INPUT                                       *
C*                                                                     *
C***********************************************************************
C
      CALL READ_NAMELIST
C
C***********************************************************************
C*                                                                     *
C*    DEFINE THE NOZZLE WALL BOUNDARY                                  *
C*                                                                     *
C***********************************************************************
C
      CALL BOUNDARY(1)
C
C***********************************************************************
C*                                                                     *
C*    GENERATE THE INITIAL VALUE LINE                                  *
C*                                                                     *
C***********************************************************************
C
      IF(IVL.EQ.0)THEN

       CALL IVLINE
C
C***********************************************************************
C*                                                                     *
C*    READ IN THE INITIAL START LINE                                   *
C*                                                                     *
C***********************************************************************
C
      ELSE

       CALL IVSTART

      ENDIF
C
C***********************************************************************
C*                                                                     *
C*    CALCULATE THE FLOW FIELD FROM THE INITIAL VALUE LINE             *
C*                                                                     *
C***********************************************************************
```

```
C
      CALL IVLFLOWFIELD
C
C*********************************************************************
C*                                                                   *
C*    DETERMINE IF THE STARTING SOLUTION REQUIRES THE CIRCULAR ARC    *
C*     WALL OR THE 2ND ORDER WALL                                    *
C*                                                                   *
C*********************************************************************
C
      IF(IVL.EQ.1 .AND. V(1).NE.0.0)THEN

       AA = ATAN(V(1)/U(1))*RAD
       RTD = 0.0
       WRITE(6,*)' '
       WRITE(6,*)' RESETTING ATTACHMENT ANGLE (AA) TO =   ',AA
       WRITE(6,*)' RESETTING DOWNSTREAM RADIUS (RTD) TO = ',RTD
       EXIT_END = .FALSE.
       DESIGNED_M = .FALSE.

      ELSE
C
C*********************************************************************
C*                                                                   *
C*    CALCULATE THE FLOW FIELD FROM THE CIRCULAR ARC THROAT CONTOUR    *
C*                                                                   *
C*********************************************************************
C
       CALL CIRCULAR_FLOWFIELD

      ENDIF
C
C*********************************************************************
C*                                                                   *
C*    CALCULATE THE FLOW FIELD FROM THE SECOND ORDER QUADRATIC WALL    *
C*                                                                   *
C*********************************************************************
C
      IF(.NOT.EXIT_END .AND. .NOT.DESIGNED_M)THEN

       CALL SECOND_ORDER_WALL_FLOWFIELD

      ENDIF
C
C*********************************************************************
C*                                                                   *
C*    FINAL C+ CHARACTERISTIC LINE CALCULATION                        *
C*                                                                   *
C*********************************************************************
C
      IF(.NOT.EXIT_END .AND. DESIGNED_M)THEN

       CALL FINAL_CHAR_LINE

      ENDIF
C
C*********************************************************************
C*                                                                   *
C*    TURNING CONTOUR REGION                                          *
C*                                                                   *
C*********************************************************************
C
      IF(.NOT.EXIT_END .AND. DESIGNED_M)THEN

       CALL TURNING_CONTOUR

      ENDIF
C
C*********************************************************************
C*                                                                   *
C*    OUTPUT THE SLOPE FILE FOR VIEWING                               *
C*                                                                   *
C*********************************************************************
C
      REWIND(3)
      IC = 0
      READ(3,*)nmlname
      DO II = 1 , 10000
       READ(3,*,END=601)nmlname
       IC = IC + 1
      ENDDO
```

```
 601  WRITE(6,*)' '
      WRITE(6,*)' WALL FILE IN: WALL.dat, AND HAS ',IC,' POINTS'

      IF(IC.GT.NII)THEN
       WRITE(6,*)' Stopping code. Redimension NII to ',IC
       STOP
      ENDIF

      OPEN(UNIT =22 , FILE = 'SLOPE.dat',  FORM = 'FORMATTED')

      REWIND(3)
      READ(3,*)nmlname
      SLOPE = 0.0
      X3    = 0.0
      DO II = 1 , IC
       READ(3,*)I4,J4,X4,Y4
       WALL_X(II) = X4
       WALL_Y(II) = Y4
       IF(II.NE.1)THEN
        SLOPE = (Y4-Y3)/(X4-X3)
        WRITE(22,*)X3,SLOPE
       ENDIF
       X3 = X4
       Y3 = Y4
      ENDDO
      WRITE(22,*)X4,SLOPE
      CLOSE(22)

      WRITE(6,*)' '
      WRITE(6,*)' WALL SLOPE FILE IN: SLOPE.dat'
C
C**********************************************************************
C*                                                                    *
C*    MAKE NEW WALL FILE - SMOOTHING                                  *
C*                                                                    *
C**********************************************************************
C
      IF(NOZ.NE.1)THEN
      OPEN(UNIT=33,FILE='WALL_SMOOTHED.dat',FORM='FORMATTED')

      CALL LEAST_SQR_FIT(WALL_X,WALL_Y,IC,IORDER,WALL_RC)

      DO II = 1 , IC
      WRITE(33,*)WALL_X(II),WALL_RC(II)
      ENDDO
      CLOSE(33)

      WRITE(6,*)' '
      WRITE(6,*)' SMOOTHED WALL FILE IN: WALL_SMOOTHED.dat'

      OPEN(UNIT =22 , FILE = 'SLOPE_SMOOTHED.dat',  FORM = 'FORMATTED')
      SLOPE = 0.0
      DO II = 1 , IC
       X4 = WALL_X(II)
       Y4 = WALL_RC(II)
       IF(II.NE.1)THEN
       SLOPE = (Y4-Y3)/(X4-X3)
       WRITE(22,*)X3,SLOPE
       ENDIF
       X3 = X4
       Y3 = Y4
      ENDDO
      WRITE(22,*)X4,SLOPE

      CLOSE(22)
C
C**********************************************************************
C*                                                                    *
C*    MAKE NEW WALL BL FILE - SMOOTHING                               *
C*                                                                    *
C**********************************************************************
C
      ICC= 0
      OPEN(UNIT=33,FILE='WALL_BOUNDARY_CORRECTED_SMOOTHED.dat',FORM='FORMATTED')
      REWIND(32)
      DO II = 1 , IC
       READ(32,231,END=232)WALL_X(II),WALL_Y(II)
       ICC = ICC + 1
 231   FORMAT(T11,E12.4,T83,E12.4)
      ENDDO
```

```
 232  CALL LEAST_SQR_FIT(WALL_X,WALL_Y,ICC,IORDER,WALL_RC)
      write(6,*)' '

      DO II = 1 , ICC
      WRITE(33,*)WALL_X(II),WALL_RC(II)
      ENDDO
      CLOSE(33)
      WRITE(6,*)' '
      WRITE(6,*)' WALL BOUNDARY LAYER CORRECTED FILE IN: WALL_BOUNDARY_CORRECTED_SMOOTHED.dat'
      ENDIF
C
C*********************************************************************
C*                                                                   *
C*    MAKE GNU PLOT FOR VIEWING DATA                                 *
C*                                                                   *
C*********************************************************************
C
      WRITE(6,*)' '
      OPEN(UNIT=55,FILE='X_Y_FUN.dem',FORM='FORMATTED')
      WRITE(55,500)
 500  FORMAT('# GNUPLOT v3.6 beta multiplot script file',/,
     >'set terminal pdf enhanced color font "Helvetica" fsize 8 size 10.5 in, 8.0 in',/,
     >'set output "X_Y_FUN.pdf"',/,
     >'set key left top box',/,
     >'set border',/,
     >'set grid',/,
     >'set style line 1 lt rgb "black"   lw 1 pt 1 ps 1',/,
     >'set style line 2 lt rgb "red"     lw 1 pt 2 ps 1',/,
     >'set style line 3 lt rgb "green"   lw 1 pt 3 ps 1',/,
     >'set style line 4 lt rgb "blue"    lw 1 pt 4 ps 1',/,
     >'set style line 5 lt rgb "orange"  lw 1 pt 5 ps 1',/,
     >'set style line 6 lt rgb "yellow"  lw 1 pt 6 ps 1',/,
     >'set style line 7 lt rgb "magenta" lw 1 pt 7 ps 1',/,
     >'show terminal ',/,
     >'set title "MOC Nozzle"',/,
     >'set xlabel "Axial Length"',/,
     >'set ylabel "Radial Length"',/,
     >'set palette defined (0 0 0 0, 1 0 0 1, 3 0 1 0, 4 1 0 0, 6 1 1 1)',/,
     >'set hidden3d')

      IF(IVL.EQ.0)THEN
      WRITE(55,501)
 501  FORMAT("plot [:] [:] \",/,
     >"'fort.13' every ::0 using 3:4 with linespoints title 'START LINE' ls 7, \",/,
     >"'fort.7' every ::0 using 3:4 with linespoints title 'INITIAL VALUE FLOW' ls 1, \",/,
     >"'fort.8' every ::0 using 3:4 with linespoints title 'CIRCULAR ARC FLOW' ls 2, \",/,
     >"'fort.9' every ::0 using 3:4 with linespoints title '2ND ORDER QUADRATIC FLOW' ls 3, \",/,
     >"'fort.10' every ::0 using 3:4 with linespoints title 'FINAL C+ CHARACTERISTIC LINE' ls 4,
\",/,
     >"'fort.11' every ::0 using 3:4 with linespoints title 'TURNING REGION' ls 5, \",/,
     >"'WALL.dat' every ::0 using 3:4 with linespoints title 'INVISCID FLOW BOUNDARY' ls 6")
      ELSE
      WRITE(55,5011)
 5011 FORMAT("plot [:] [:] \",/,
     >"'fort.7' every ::0 using 3:4 with linespoints title 'INITIAL VALUE LINE' ls 1, \",/,
     >"'fort.9' every ::0 using 3:4 with linespoints title '2ND ORDER QUADRATIC WALL' ls 3, \",/,
     >"'fort.10' every ::0 using 3:4 with linespoints title 'FINAL C+ CHARACTERISTIC LINE' ls 4,
\",/,
     >"'fort.11' every ::0 using 3:4 with linespoints title 'TURNING REGION' ls 5, \",/,
     >"'WALL.dat' every ::0 using 3:4 with linespoints title 'INVISCID FLOW BOUNDARY' ls 6")
      ENDIF

      IF(IVL.EQ.0)THEN
      WRITE(55,502)
 502  FORMAT("plot [:] [:] \",/,
     >"'fort.7' every ::0 using 3:4 with lines title 'INITIAL VALUE LINE' ls 1, \",/,
     >"'fort.8' every ::0 using 3:4 with lines title 'CIRCULAR ARC THROAT' ls 2, \",/,
     >"'fort.9' every ::0 using 3:4 with lines title '2ND ORDER QUADRATIC WALL' ls 3, \",/,
     >"'fort.10' every ::0 using 3:4 with lines title 'FINAL C+ CHARACTERISTIC LINE' ls 4, \",/,
     >"'fort.11' every ::0 using 3:4 with lines title 'TURNING REGION' ls 5, \",/,
     >"'WALL.dat' every ::0 using 3:4 with lines title 'INVISCID FLOW BOUNDARY' ls 6")
      ELSE
      WRITE(55,5021)
 5021 FORMAT("plot [:] [:] \",/,
     >"'fort.7' every ::0 using 3:4 with lines title 'INITIAL VALUE LINE' ls 1, \",/,
     >"'fort.9' every ::0 using 3:4 with lines title '2ND ORDER QUADRATIC WALL' ls 3, \",/,
     >"'fort.10' every ::0 using 3:4 with lines title 'FINAL C+ CHARACTERISTIC LINE' ls 4, \",/,
     >"'fort.11' every ::0 using 3:4 with lines title 'TURNING REGION' ls 5, \",/,
     >"'WALL.dat' every ::0 using 3:4 with lines title 'INVISCID FLOW BOUNDARY' ls 6")
      ENDIF
```

```
      IF(IVL.EQ.0)THEN
      WRITE(55,503)
 503  FORMAT("set pm3d map",/,'set title "Mach Number"',/,"splot [:] [:] [:] \",/,
     >"'fort.7' every ::0 using 3:4:7 notitle ls 1, \",/,
     >"'fort.8' every ::0 using 3:4:7 notitle ls 2, \",/,
     >"'fort.9' every ::0 using 3:4:7 notitle ls 3, \",/,
     >"'fort.11' every ::0 using 3:4:7 notitle ls 5")
      ELSE
      WRITE(55,5031)
 5031 FORMAT("set pm3d map",/,'set title "Mach Number"',/,"splot [:] [:] [:] \",/,
     >"'fort.7' every ::0 using 3:4:7 notitle ls 1, \",/,
     >"'fort.9' every ::0 using 3:4:7 notitle ls 3, \",/,
     >"'fort.11' every ::0 using 3:4:7 notitle ls 5")
      ENDIF

      IF(IVL.EQ.0)THEN
      WRITE(55,504)
 504  FORMAT("set pm3d map",/,'set title "Axial Velocity"',/,"splot [:] [:] [:] \",/,
     >"'fort.7' every ::0 using 3:4:5 notitle ls 1, \",/,
     >"'fort.8' every ::0 using 3:4:5 notitle ls 2, \",/,
     >"'fort.9' every ::0 using 3:4:5 notitle ls 3, \",/,
     >"'fort.11' every ::0 using 3:4:5 notitle ls 5")
      ELSE
      WRITE(55,5041)
 5041 FORMAT("set pm3d map",/,'set title "Axial Velocity"',/,"splot [:] [:] [:] \",/,
     >"'fort.7' every ::0 using 3:4:5 notitle ls 1, \",/,
     >"'fort.9' every ::0 using 3:4:5 notitle ls 3, \",/,
     >"'fort.11' every ::0 using 3:4:5 notitle ls 5")
      ENDIF

      IF(IVL.EQ.0)THEN
      WRITE(55,505)
 505  FORMAT("set pm3d map",/,'set title "Radial Velocity"',/,"splot [:] [:] [:] \",/,
     >"'fort.7' every ::0 using 3:4:6 notitle ls 1, \",/,
     >"'fort.8' every ::0 using 3:4:6 notitle ls 2, \",/,
     >"'fort.9' every ::0 using 3:4:6 notitle ls 3, \",/,
     >"'fort.11' every ::0 using 3:4:6 notitle ls 5")
      ELSE
      WRITE(55,5051)
 5051 FORMAT("set pm3d map",/,'set title "Radial Velocity"',/,"splot [:] [:] [:] \",/,
     >"'fort.7' every ::0 using 3:4:6 notitle ls 1, \",/,
     >"'fort.9' every ::0 using 3:4:6 notitle ls 3, \",/,
     >"'fort.11' every ::0 using 3:4:6 notitle ls 5")
      ENDIF

      IF(IVL.EQ.0)THEN
      WRITE(55,506)
 506  FORMAT("set pm3d map",/,
     >'set title "Flow Angle"',/,"splot [:] [:] [:] \",/,
     >"'fort.7' every ::0 using 3:4:9 notitle ls 1, \",/,
     >"'fort.8' every ::0 using 3:4:9 notitle ls 2, \",/,
     >"'fort.9' every ::0 using 3:4:9 notitle ls 3, \",/,
     >"'fort.11' every ::0 using 3:4:9 notitle ls 5")
      ELSE
      WRITE(55,5061)
 5061 FORMAT("set pm3d map",/,
     >'set title "Flow Angle"',/,"splot [:] [:] [:] \",/,
     >"'fort.7' every ::0 using 3:4:9 notitle ls 1, \",/,
     >"'fort.9' every ::0 using 3:4:9 notitle ls 3, \",/,
     >"'fort.11' every ::0 using 3:4:9 notitle ls 5")
      ENDIF

      IF(NOZ.EQ.1)THEN
      WRITE(55,507)
 507  FORMAT('set title "Wall Profile"',/,"plot [:] [:] \",/,
     >"'WALL.dat' every ::0 using 3:4 with lines title 'Wall' ls 1, \",/,
     >"'WALL_BOUNDARY_CORRECTED.dat' every ::0 using 3:9 with lines title 'Wall w/BL Correction'
ls 3")
      ELSE
      WRITE(55,5071)
 5071 FORMAT('set title "Wall Profile"',/,"plot [:] [:] \",/,
     >"'WALL.dat' every ::0 using 3:4 with lines title 'Wall' ls 1, \",/,
     >"'WALL_SMOOTHED.dat' every ::0 using 1:2 with lines title 'Wall - Smoothed' ls 2, \",/,
     >"'WALL_BOUNDARY_CORRECTED.dat' every ::0 using 3:9 with lines title 'Wall w/BL Correction'
ls 3, \",/,
     >"'WALL_BOUNDARY_CORRECTED_SMOOTHED.dat' every ::0 using 1:2 with lines title 'Wall w/BL
Correction - Smoothed' ls 4")
      ENDIF

      IF(NOZ.NE.1)THEN
      WRITE(55,508)
```

```
 508  FORMAT('set ylabel "Surface Slope (Degrees)"',/,'set title "Wall Surface Slope"',/,
     >"plot [:] [:] \",/,
     >"'SLOPE.dat' every ::0 using ($1):(atan($2)/pi*180.) with linespoints title 'Wall Slope' ls
1, \",/,
     >"'SLOPE_SMOOTHED.dat' every ::0 using ($1):(atan($2)/pi*180.) with linespoints title 'Wall
Slope - Smoothed' ls 2")
      ENDIF

      WRITE(6,*)' '
      WRITE(6,*)' Writing gnuplot X_Y_FUN.dem'
      WRITE(6,*)' '
      WRITE(6,*)'  Type: gnuplot X_Y_FUN.dem ; open X_Y_FUN.pdf'
      STOP
C
C*********************************************************************
C*                                                                   *
C*    END OF LINE                                                    *
C*                                                                   *
C*********************************************************************
C
      END
      SUBROUTINE AXIS
C...
C...  Points 1 and 3 are known
C...  Point 4 is located on the axis
C...  Point 3 is located on the axis
C...  The C- characteristic goes through points 1 and 4
C...  The C+ characteristic goes through points 3 and 1
C...  The streamline        goes through points 3 and 4
C...
C...                        1
C...                        *
C...                       * *
C...                 C+  *   *  C-
C...                    *     *
C...                   *       *
C...  Solution Point  3*  AXIS   *4 Solution Point
C...  ------------------------------------------

C...
C... CALCULATE THE SOLUTION AT AN AXIS OF SYMMETRY POINT
C...  POINTS 1 AND 3 ARE KNOWN, SOLVE FOR POINT 4 ALONG AXIS USING CHARA-
C...
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C...
C... CALCULATE THE COEFFICIENTS FOR THE PREDICTOR
C...
      ITER = 0

      Q4 = Q3
      P4 = P3
      R4 = R3

      CALL THERMO (Q1,P1,R1,T,C,M)

      LM = TAN(A1-ASIN(1./M))
      QM = GC*SQRT(M**2-1.)/(R1*Q1**2)
      SM = DELTA*SIN(A1)/(Y1*M*COS(A1-ASIN(1./M)))

      A4 = 0.0  !KNOWN
      Y4 = 0.0  !KNOWN
C...
C... SOLUTION
C...
 10   X4 = X1-Y1/LM

      TM = -SM*(X4-X1)+QM*P1-A1

      P0 = 0.5*(P3+P4)
      R8 = 0.5*(R3+R4)
      Q0 = 0.5*(Q3+Q4)
      R0 = R8*Q0/GC

      CALL THERMO (Q0,P0,R8,T,C,M)

      A0 = C**2/GC
      T01 = R0*Q3+P3
      T02 = P3-A0*R3

      P4 = (TM-A4)/QM
```

```
      Q4 = (T01-P4)/R0
      R4 = (P4-T02)/A0
C...
C... TEST FOR CONVERGENCE
C...

c     WRITE(6,*)' '
c     WRITE(6,100)' R0                    = ',R0
c     WRITE(6,100)' A0                    = ',A0
c     WRITE(6,100)' T01                   = ',T01
c     WRITE(6,100)' T02                   = ',T02
c     WRITE(6,100)' Q-                    = ',QM
c     WRITE(6,100)' S-                    = ',SM
c     WRITE(6,100)' T-                    = ',TM
 100  FORMAT(A,1P2E15.5)

      IF(ITER.EQ.ICOR) RETURN
      IF(ITER.EQ.0)    GOTO 20

      IF((ABS(X4-XD).GT.E1)    .OR.  (ABS(Q4-QD).GT.E4*QD)) GOTO 20
      IF((ABS(P4-PD).GT.E2*PD) .AND.  (ABS(R4-RD).GT.E3*RD)) RETURN
C...
C... CALCULATE THE COEFFICIENTS FOR THE CORRECTOR
C...
 20   ITER = ITER + 1

      XD = X4
      PD = P4
      RD = R4
      QD = Q4

      P0 = 0.5*(P1+P4)
      R0 = 0.5*(R1+R4)
      Q0 = 0.5*(Q1+Q4)
      A0 = 0.5*(A1+A4)
      Y0 = 0.5*(Y1+Y4)

      CALL THERMO (Q0,P0,R0,T,C,M)

      LM = TAN(A0-ASIN(1.0/M))
      SM = DELTA*SIN(A0)/(Y0*M*COS(A0-ASIN(1.0/M)))
      QM = GC*SQRT(M**2-1.0)/(R0*Q0**2)

      GOTO 10

      END
      SUBROUTINE INVERSE_WALL_POINT
C...
C...  Points 1 and 3 are known
C...  Point 4 is located on the wall
C...  Point 3 is located on the wall
C...  Point 2 is located between points 1 and 3
C...  The C- characteristic goes through points 1 and 3
C...  The C+ characteristic goes through points 2 and 4
C...  The streamline        goes through points 3 and 4
C...
C...                ##
C...   WALL      ##
C...          ## *4
C...       ##   *
C...    ##     *
C... ##       * C+
C...  3*     *
C...    *   *
C...  C- * *
C...     2*
C...       *
C...        * 1

C...
C... Calculate the solution at an inverse wall point
C...
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C...
C... CALCULATE THE COEFFICIENTS FOR THE PREDICTOR
C...
      ITER = 0

      LM = (Y3-Y1)/(X3-X1)
```

```
      P4 = 0.5*(P1+P3)
      R4 = 0.5*(R1+R3)
      Q4 = 0.5*(Q1+Q3)

      XA = X3

      P2 = P4
      R2 = R4
      Q2 = Q4
      A2 = 0.5*(A1+A3)
C...
C... DETERMINE THE LOCATION AND PROPERTIES OF POINT 2
C...
 10   Q0 = 0.5*(Q2+Q4)
      A0 = 0.5*(A2+A4)
      P0 = 0.5*(P2+P4)
      R8 = 0.5*(R2+R4)

      CALL THERMO (Q0,P0,R8,T,C,M)

      LP = TAN(A0+ASIN(1./M))

      X2 = (Y4-Y1+lm*x1-lp*x4)/(lm-lp)
      Y2 = Y4+LP*(X2-X4)

      D  = (X2-X1)/(X3-X1)

      Q2 = Q1+D*(Q3-Q1)
      A2 = A1+D*(A3-A1)
      P2 = P1+D*(P3-P1)
      R2 = R1+D*(R3-R1)

      IF(ITER.EQ.0)GOTO 15

      P4 = P2
      R4 = R2
      Q4 = Q2

 15   IF(ABS(X2-XA).LT.ERROR) GOTO 20

      XA = X2

      GOTO 10
************************************************

 20   P0 = 0.5*(P2+P4)
      R8 = 0.5*(R2+R4)
      Q0 = 0.5*(Q2+Q4)
      A0 = 0.5*(A2+A4)
      Y0 = 0.5*(Y2+Y4)

      CALL THERMO (Q0,P0,R8,T,C,M)

      LP = TAN(A0+ASIN(1.0/M))
      QP = GC*SQRT(M**2-1.0)/(R8*Q0**2)
      SP = DELTA*SIN(A0)/(Y0*M*COS(A0+ASIN(1.0/M)))
      TP = -SP*(X4-X2)+QP*P2+A2

      IF(ITER.GT.0)GOTO 25

      P4 = P3
      Q4 = Q3
      R4 = R3

 25   P0 = 0.5*(P3+P4)
      R8 = 0.5*(R3+R4)
      Q0 = 0.5*(Q3+Q4)

      R0 = R8*Q0/GC

      CALL THERMO (Q0,P0,R8,T,C,M)

      A0 = C**2/GC
      T01=R0*Q3+P3
      T02=P3-A0*R3
C...
C... CALCULATE THE PROPERITIES AT POINT 4
C...
      P4 = (TP-A4)/QP
      Q4 = (T01-P4)/R0
      R4 = (P4-T02)/A0
```

```
C...
C... TEST FOR CONVERGENCE
C...
c     WRITE(6,*)' '
c     WRITE(6,100)' R0                = ',R0
c     WRITE(6,100)' A0                = ',A0
c     WRITE(6,100)' T01               = ',T01
c     WRITE(6,100)' T02               = ',T02
c     WRITE(6,100)' Q+                = ',QP
c     WRITE(6,100)' S+                = ',SP
c     WRITE(6,100)' T+                = ',TP
c     WRITE(6,100)' Q-                = ',QM
c     WRITE(6,100)' S-                = ',SM
c     WRITE(6,100)' T-                = ',TM
      IF(ITER.EQ.ICOR) RETURN
      IF(ITER.EQ.0)    GOTO 40

      IF((ABS(P4-PD).GT.E2*PD) .OR.  (ABS(R4-RD).GT.E3*RD)) GOTO 40
      IF((ABS(Q4-QD).GT.E4*QD) ) RETURN
C...
C... CALCULATE THE COEFFICIENTS FOR THE CORRECTOR
C...
 40   ITER = ITER + 1

      PD = P4
      RD = R4
      QD = Q4

 100  FORMAT(A,1P2E15.5)
      GOTO 10

      END
      SUBROUTINE SPLINT(XA,YA,Y2A,N,X,Y)
      DIMENSION XA(N),YA(N),Y2A(N)
      KLO=1
      KHI=N
1     IF (KHI-KLO.GT.1) THEN
        K=(KHI+KLO)/2
        IF(XA(K).GT.X)THEN
          KHI=K
        ELSE
          KLO=K
        ENDIF
      GOTO 1
      ENDIF
      H=XA(KHI)-XA(KLO)
      IF (H.EQ.0.) PAUSE 'Bad XA input.'
      A=(XA(KHI)-X)/H
      B=(X-XA(KLO))/H
      Y=A*YA(KLO)+B*YA(KHI)+
     *      ((A**3-A)*Y2A(KLO)+(B**3-B)*Y2A(KHI))*(H**2)/6.
      RETURN
      END
      SUBROUTINE BANNER(ASTR)

      CHARACTER*132 ASTR,C2,C3
      CHARACTER*1 C1
      CHARACTER*2 C4

      C1 ='C'
      C2 ='C********************************************************************'
      C3 ='C**                                                                **'
      C4 ='**'

      WRITE(6,FMT='(A)')C1
      WRITE(6,FMT='(A)')C2
      WRITE(6,FMT='(A)')C3
      WRITE(6,FMT='(A)')C3(1:5)//ASTR(1:65)//C4(1:2)
      WRITE(6,FMT='(A)')C3
      WRITE(6,FMT='(A)')C2
      WRITE(6,FMT='(A)')C1

      RETURN
      END
      SUBROUTINE IVLFLOWFIELD
C
C***********************************************************************
C*                                                                     *
C*    CALCULATE THE FLOW FIELD FROM THE INITIAL VALUE LINE             *
C*                                                                     *
C***********************************************************************
```

```
C
C
C**********************************************************************
C*                                                                    *
C*      TERMINOLOGY FOR SUPERSONIC FLOW METHOD OF CHARACTERISTICS     *
C*                                                                    *
C* CONTROL VARIABLE:                                                  *
C* ----------------                                                   *
C*                                                                    *
C* DELTA  = '0' FOR PLANER FLOW                                       *
C*        = '1' AXISYMMETRIC FLOW                                     *
C* ICOR   = NUMBER OF APPLICATIONS OF THE CORRECTOR DESIRED           *
C* E1     = CONVERGENCE TOLERANCE FOR LOCATION, M (IN)                *
C* E2     = CONVERGENCE TOLERANCE FOR VELOCITY, M/S (FT/SEC)          *
C* GC     = 1.0 M-KG/N-S^2 OR 32.174 FT-LBM/LBF-S^2                   *
C* GL     = 1.0 M^2/M^2 OR 144.0 IN^2/FT^2                            *
C* ST     = 0.0 EVEN SPACE STARTLINE, 1.0 QUADRATIC SPACED            *
C*                                                                    *
C* GAS THERMODYNAMIC PROPERTIES & STAGNATION PROPERTIES:              *
C* ----------------------------------------------------               *
C*                                                                    *
C* G      = RATIO OF SPECIFIC HEATS                                   *
C* RG     = GAS CONSTANT, J/KG-K (FT-LBF/LBM-R)                       *
C* TS     = STAGNATION TEMPERATURE, K (R)                             *
C* PS     = STAGNATION PRESSURE, N/M^2 (LBF/IN^2)                     *
C* PA     = AMBIENT PRESSURE, N/M^2 (LBF/IN^2)                        *
C*                                                                    *
C* FLOW FIELD PROPERTIES:                                             *
C* ---------------------                                              *
C*                                                                    *
C* X      = AXIAL COORDINATE, M (IN)                                  *
C* Y      = RADIAL COORDINATE, M (IN)                                 *
C* U      = AXIAL VELOCITY, M/S (FT/S)                                *
C* V      = RADIAL VELOCITY, M/S (FT/S)                               *
C* Q      = VELOCITY MAGNITUDE, M/W (FT/S)                            *
C* A      = FLOW ANGLE, RAD                                           *
C* P      = STATIC PRESSURE, N/M^2 (LBF/IN^2)                         *
C* R      = STATIC DENSITY, KG/M^3 (LBM/FT^3)                         *
C* T      = STATIC TEMPERATURE, K (R)                                 *
C* C      = SPEED OF SOUND, M/S (FT/S)                                *
C* M      = MACH NUMBER                                               *
C* EMD    = DESIGN MACH NUMBER                                        *
C* 1,2,3, = DENOTES PROPERTIES AT POINTS                              *
C*                                                                    *
C* TERMINOLOGY EMPLOYED:                                              *
C* --------------------                                               *
C*                                                                    *
C* L      = TAN(THETA+-ALPHA)                                         *
C* Q      = (U^2-C^2), M^2/S^2 (FT^2/S^2)                             *
C* R      = 2UV-L(U^2-C^2) M^2/S^2 (FT^2/S^2)                         *
C* S      = DELTA*C^2*V/Y, M^2/S^3 (FT^2/SEC^3-IN)                    *
C* T      = S*DEL(X)+Q*U+R*V, M^3/S^3 (FT^3/S^3)                      *
C* +/-    = DENOTES + OR - CHARACTERISTIC CURVE                       *
C*                                                                    *
C**********************************************************************
C
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C
C*********************************************************************
C**                                                                 **
C**   Function Statements                                           **
C**                                                                 **
C*********************************************************************
C
      TM(B) = B / TZERO
      ZMUIFD(B) = ZMZERO*TM(B)**1.5*((1.+CONSU)/(TM(B)+CONSU))
      RE(B) = R(1) * Q(1) * X(1) / ZMUIFD(B)
      BL(B) = RE(B)**(1./8.)
C
C**********************************************************************
C*                                                                    *
C*    OUTPUT THE HEADER                                               *
C*                                                                    *
C**********************************************************************
C
      WRITE(6,*)' '
      ASTR = 'CALCULATE THE FLOWFIELD FROM THE INITIAL START LINE'
      CALL BANNER(ASTR)

      WRITE(3,1010)
```

```
      WRITE(4,1010)
C
C*********************************************************************
C*                                                                   *
C*    LOOP OVER THE NUMBER OF START LINE POINTS - I                  *
C*                                                                   *
C*********************************************************************
C
      DO 80 I = 1 , NI

C...FIND THE NUMBER OF RADIAL POINTS FOR EACH I LINE

       J1 = NI - I + 1         !Lower bound on J - initial value line
       J2 = J1 + 2.*(I*1.-1.)  !Upper bound on J - centerline value

       WRITE(6,1010)
       WRITE(7,*)' '
C
C*********************************************************************
C*                                                                   *
C*    RESET THE MASS FLOW RATE                                       *
C*                                                                   *
C*********************************************************************
C
       MFRT = 0.0
       KW = 1
C
C*********************************************************************
C*                                                                   *
C*    LOOP OVER THE NUMBER OF RADIAL POINTS FOR EACH I LINE          *
C*                                                                   *
C*********************************************************************
C
       DO 81 J = J1 , J2        !recall that as j increases the physical
                                ! location moves away from startline
                                ! towards the centerline

        IF(J.EQ.J1)THEN         !on initial value line data
         K = 2*NI - 1 + I
         CALL MOVE(4,K)         !move current point into point 4
        ELSEIF(J.EQ.J2)THEN     !on centerline
         CALL MOVE(1,J-1)       !move the upper point into point 1
         CALL MOVE(6,0)         !move the old P2 point from INTERIOR_POINT into P3
         CALL AXIS
        ELSE                    !on interior point
         CALL MOVE(1,J-1)       !move the upper point into point 1
         CALL MOVE(2,J)         !move the current point into point 2
         CALL INTERIOR_POINT
        ENDIF

        CALL MOVE(5,J)          !save point 4 into current array

        AP = A4*RAD
        PP = P4 /GL
C
C*********************************************************************
C*                                                                   *
C*    CALCULATE THE MASS FLOW RATE                                   *
C*                                                                   *
C*********************************************************************
C
       IF(J.EQ.J1)THEN
        AREA_X = 0.0
        AREA_Y = 0.0
        DEL_X  = 0.0
        DEL_Y  = 0.0
        R_AVE  = 0.0
        U_AVE  = 0.0
        V_AVE  = 0.0
       ELSE
        DEL_X  = ABS(X4-X(J-1))
        DEL_Y  = ABS(Y4-Y(J-1))
        AREA_X = PI * ABS(Y4**2 - Y(J-1)**2 ) * DELTA + (1-DELTA)*ABS(Y4-Y(J-1))
        AREA_Y = 2.0 * PI * (Y4 + Y(J-1))/2.0*DEL_X * DELTA + (1-DELTA)*DEL_X
        R_AVE  = (R4 + R(J-1))/2.0
        U_AVE  = (U4 + U(J-1))/2.0
        V_AVE  = (V4 + V(J-1))/2.0
       ENDIF

        MFR     = R_AVE*(U_AVE*AREA_X+V_AVE*AREA_Y)
        MFRT    = MFR + MFRT
```

```
        ERROR_M = (MFRT-MDOT)/MDOT
C*
C***********************************************************************
C*                                                                     *
C*    OUTPUT THE FLOW VARIABLES                                        *
C*                                                                     *
C***********************************************************************
C*
        IF (KW.EQ.1)THEN
         WRITE(6,1020)I,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
         WRITE(7,1020)I,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4

         IF(J.EQ.1)WRITE(3,1020)I,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4

IF(J.EQ.1)WRITE(32,1020)I,J,X4,Y4,TM(T4),ZMUIFD(T4),RE(T4),BL(T4),Y4+BL_SCALE*0.1404*((1.+X4)**0.
125-1.0)
         IF(J.EQ.J2)WRITE(4,1020)I,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
         IF(J.EQ.J2 .AND. I.EQ.NI)write(6,*)' MASS FLUX = ',MFRT,
     >   ' PERCENT ERROR = ',ERROR_M*100.
        ENDIF
C*
C***********************************************************************
C*                                                                     *
C*    PLACE THE FLOW VARIABLES INTO STORAGE                            *
C*                                                                     *
C***********************************************************************
C*
         XY(I,J,1)   = X4
         XY(I,J,2)   = Y4
         FLOW(I,J,1) = U4
         FLOW(I,J,2) = V4
         FLOW(I,J,3) = M4
         FLOW(I,J,4) = Q4
         FLOW(I,J,5) = A4
         FLOW(I,J,6) = P4
         FLOW(I,J,7) = R4
         FLOW(I,J,8) = T4
c        JMAX = MAX(J,JMAX)
C*
C***********************************************************************
C*                                                                     *
C*    CHECK FOR CROSSING LINES                                         *
C*                                                                     *
C***********************************************************************
C*
        IF(I.GT.2 .AND. J.GE.J1+1)THEN

         P1_X = XY(I,J,1)
         P1_Y = XY(I,J,2)
         P2_X = XY(I,J-1,1)
         P2_Y = XY(I,J-1,2)

         DO JJ = J1+2 , J2-1

          P3_X = XY(I-1,JJ,1)
          P3_Y = XY(I-1,JJ,2)
          P4_X = XY(I-1,JJ-1,1)
          P4_Y = XY(I-1,JJ-1,2)

          CALL LINE_INTERSECTION(P1_X,P1_Y,P2_X,P2_Y,P3_X,
     >    P3_Y,P4_X,P4_Y,H_FACTOR,G_FACTOR,IX,IY)

          IF(H_FACTOR.GE.0.0 .AND. H_FACTOR.LE.1.0 .AND.
     >       G_FACTOR.GE.0.0 .AND. G_FACTOR.LE.1.0 )THEN

           WRITE(6,*)' [31mCHARACTERISTICS INTERSECTION DEDECTED![0m'

C...interpolate for new point
           X4 = (1.0 - G_FACTOR)*P1_X + G_FACTOR*P2_X
           Y4 = (1.0 - G_FACTOR)*P1_Y + G_FACTOR*P2_Y
           U4 = (1.0 - G_FACTOR)*U4 + G_FACTOR*FLOW(I,J-1,1)
           V4 = (1.0 - G_FACTOR)*V4 + G_FACTOR*FLOW(I,J-1,2)
           M4 = (1.0 - G_FACTOR)*M4 + G_FACTOR*FLOW(I,J-1,3)
           Q4 = (1.0 - G_FACTOR)*Q4 + G_FACTOR*FLOW(I,J-1,4)
           A4 = (1.0 - G_FACTOR)*A4 + G_FACTOR*FLOW(I,J-1,5)
           P4 = (1.0 - G_FACTOR)*P4 + G_FACTOR*FLOW(I,J-1,6)
           R4 = (1.0 - G_FACTOR)*R4 + G_FACTOR*FLOW(I,J-1,7)
           T4 = (1.0 - G_FACTOR)*T4 + G_FACTOR*FLOW(I,J-1,8)
C*
C***********************************************************************
C*                                                                     *
```

```
C*    PLACE THE FLOW VARIABLES INTO STORAGE                              *
C*                                                                        *
C************************************************************************
C*
           XY(I,J,1)    = X4
           XY(I,J,2)    = Y4
           FLOW(I,J,1) = U4
           FLOW(I,J,2) = V4
           FLOW(I,J,3) = M4
           FLOW(I,J,4) = Q4
           FLOW(I,J,5) = A4
           FLOW(I,J,6) = P4
           FLOW(I,J,7) = R4
           FLOW(I,J,8) = T4

           X(J) = X4
           Y(J) = Y4
           P(J) = P4
           R(J) = R4
           U(J) = U4
           V(J) = V4
           Q(J) = Q4
           A(J) = A4

           BACKSPACE(7)
           WRITE(7,1020)I,J,X4,Y4,U4,V4,M4,Q4,A4*RAD,P4/GL,R4,T4
           WRITE(6,1020)I,J,X4,Y4,U4,V4,M4,Q4,A4*RAD,P4/GL,R4,T4

           DO JJJ = J+1 , J2
            XY(I,JJJ,1)    = XY(I-1,JJJ-1,1)
            XY(I,JJJ,2)    = XY(I-1,JJJ-1,2)
            FLOW(I,JJJ,1) = FLOW(I-1,JJJ-1,1)
            FLOW(I,JJJ,2) = FLOW(I-1,JJJ-1,2)
            FLOW(I,JJJ,3) = FLOW(I-1,JJJ-1,3)
            FLOW(I,JJJ,4) = FLOW(I-1,JJJ-1,4)
            FLOW(I,JJJ,5) = FLOW(I-1,JJJ-1,5)
            FLOW(I,JJJ,6) = FLOW(I-1,JJJ-1,6)
            FLOW(I,JJJ,7) = FLOW(I-1,JJJ-1,7)
            FLOW(I,JJJ,8) = FLOW(I-1,JJJ-1,8)
            X(JJJ) = XY(I,JJJ,1)
            Y(JJJ) = XY(I,JJJ,2)
            P(JJJ) = FLOW(I,JJJ,6)
            R(JJJ) = FLOW(I,JJJ,7)
            U(JJJ) = FLOW(I,JJJ,1)
            V(JJJ) = FLOW(I,JJJ,2)
            Q(JJJ) = FLOW(I,JJJ,4)
            A(JJJ) = FLOW(I,JJJ,5)
            WRITE(6,1020)I,JJJ,XY(I,JJJ,1),XY(I,JJJ,2),FLOW(I,JJJ,1),FLOW(I,JJJ,2),
     >
FLOW(I,JJJ,3),FLOW(I,JJJ,4),FLOW(I,JJJ,5)*RAD,FLOW(I,JJJ,6)/GL,FLOW(I,JJJ,7),FLOW(I,JJJ,8)
            WRITE(7,1020)I,JJJ,XY(I,JJJ,1),XY(I,JJJ,2),FLOW(I,JJJ,1),FLOW(I,JJJ,2),
     >
FLOW(I,JJJ,3),FLOW(I,JJJ,4),FLOW(I,JJJ,5)*RAD,FLOW(I,JJJ,6)/GL,FLOW(I,JJJ,7),FLOW(I,JJJ,8)
           ENDDO

           IF(SMOOTH)THEN
            J_START = J1
            J_END   = J2
            I_LINE = I
            I_FILE = 7
            CALL SMOOTH_PROFILE(J_START,J_END,I_LINE,0,I_FILE)
           ENDIF

           GOTO 80
          ENDIF

         ENDDO
C*
C************************************************************************
C*                                                                        *
C*    END OF LINE CROSSING CHECK                                          *
C*                                                                        *
C************************************************************************
C*
        ENDIF
C*
C************************************************************************
C*                                                                        *
C*    END OF DOWNWARD J LOOP                                              *
C*                                                                        *
C************************************************************************
```

```
C*
 81   CONTINUE
C*
C***********************************************************************
C*                                                                     *
C*    END OF I MACHING INDEX                                           *
C*                                                                     *
C***********************************************************************
C*
 80   CONTINUE
C
C***********************************************************************
C*                                                                     *
C*    SMOOTH THE PROFILE                                               *
C*                                                                     *
C***********************************************************************
C
      IF(SMOOTH)THEN
       DO LL = 1 , 2*NI-1
        BACKSPACE(7)
       ENDDO
       J_START = 1
c      J_END   = 1 + 2.*(NI*1.-1.)
       J_END   = 2*NI-1
       I_LINE = NI
       I_FILE = 7
       CALL SMOOTH_PROFILE(J_START,J_END,I_LINE,0,I_FILE)
      ENDIF
C
C***********************************************************************
C*                                                                     *
C*    FORMAT STATEMENTS                                                *
C*                                                                     *
C***********************************************************************
C
c1010 FORMAT(5X,'I',4X,'J',6X,'X',9X'Y',9X,'U',9X,'V',9X,'M',9X,'Q',
c    >9X,'A',9X,'P',12X,'R',12X,'T',/)
 1010 FORMAT(/,'    I    J ','   X         ', '      Y      ',' U VELOCITY ', ' V VELOCITY ',
     >'MACH NUMBER ', 'VELOCITY MAG',' FLOW ANGLE ',' PRESSURE   ','  DENSITY   ',' TEMPERATURE')
 1020 FORMAT(2I5,1P10E12.4)
      RETURN
C
C***********************************************************************
C*                                                                     *
C*    END OF LINE                                                      *
C*                                                                     *
C***********************************************************************
C
      END
      SUBROUTINE MOVE(I,J)
C
C**********************************************************************
C**                                                                  **
C**   SUBROUTINE MOVE WILL LOAD THE J INDEX FLOW VARIABLE INTO THE   **
C**   CURRENT I POINT.                                               **
C**                                                                  **
C**********************************************************************
C*
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C
C**********************************************************************
C**                                                                  **
C**   GO TO THE CORRECT MOVE LOCATION.                               **
C**                                                                  **
C**********************************************************************
C*
      GOTO (10,20,30,40,50,60) I
C
C**********************************************************************
C**                                                                  **
C**   LOAD CURRENT ARRAY INTO POINT 1.                               **
C**                                                                  **
C**********************************************************************
C*
 10   X1 = X(J)
      Y1 = Y(J)
      P1 = P(J)
      R1 = R(J)
      Q1 = Q(J)
```

```
      A1 = A(J)

      RETURN
C
C**********************************************************************
C**                                                                  **
C**   LOAD CURRENT ARRAY INTO POINT 2.                               **
C**                                                                  **
C**********************************************************************
C*
 20   X2 = X(J)
      Y2 = Y(J)
      P2 = P(J)
      R2 = R(J)
      Q2 = Q(J)
      A2 = A(J)

      RETURN
C
C**********************************************************************
C**                                                                  **
C**   LOAD CURRENT ARRAY INTO POINT 3.                               **
C**                                                                  **
C**********************************************************************
C*
 30   X3 = X(J)
      Y3 = Y(J)
      P3 = P(J)
      R3 = R(J)
      Q3 = Q(J)
      A3 = A(J)

      RETURN
C
C**********************************************************************
C**                                                                  **
C**   LOAD CURRENT ARRAY INTO POINT 4.                               **
C**                                                                  **
C**********************************************************************
C*
 40   X4 = X(J)
      Y4 = Y(J)
      P4 = P(J)
      R4 = R(J)
      Q4 = Q(J)
      A4 = A(J)

      RETURN
C
C**********************************************************************
C**                                                                  **
C**   LOAD POINT 4 INTO THE ARRAY.                                   **
C**                                                                  **
C**********************************************************************
C*
 50   X(J) = X4
      Y(J) = Y4
      P(J) = P4
      R(J) = R4
      Q(J) = Q4
      A(J) = A4
      U4 = Q4*COS(A4)
      V4 = Q4*SIN(A4)
      U(J) = U4
      V(J) = V4

      CALL THERMO(Q4,P4,R4,T4,C,M4)

      RETURN
C
C**********************************************************************
C**                                                                  **
C**   LOAD POINT 2 INTO POINT 3.                                     **
C**                                                                  **
C**********************************************************************
C*

 60   X3 = X2
      Y3 = Y2
      P3 = P2
      R3 = R2
```

```
      Q3 = Q2
      A3 = A2

      RETURN
C
C*********************************************************************
C**                                                                 **
C**   END OF SUBROUTINE                                             **
C**                                                                 **
C*********************************************************************
C*
      END
      SUBROUTINE THERMO(Q0,P0,R0,T,C,M)
C...
C... CALCULATE T - Temp, C - Speed of Sound, AND M - MACH NUMBER
C... FOR GIVEN Q - Velocity, P - Pressure, AND R - Density
C...
C
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'

      T = P0 / (RG*R0)

      C = SQRT(G*GC*RG*T)

      M = Q0 / C

      RETURN
      END
      SUBROUTINE BOUNDARY(INITIAL)
C
C**********************************************************************
C*
C... FOR A NOZZLE HAVING A CIRCULAR ARC THROAT JOINED TANGENTIALLY TO
c... A WALL HAVING A CONTOUR DESCRIBED BY A SECOND ORDER POLYNOMIAL -
C... LOCATE THE INTERSECTION OF A LEFT RUNNING CHARACTERISTIC
c... WITH A SECOND ORDER QUADRATIC WALL CONTOUR
c...
c... INPUT:
C...        YT  is the throat radius
C...        RTD is the radius of curvature of throat above YT
C...        AA  is angle for RTD
C...        AE  is angle at lip
C...        XE  is the exit location
C...
C...
C...        |*
C...        | * <- Angle AA      - <- Exit Location XE with angle AE
C...        |  *            -
C...    RTD |   *         -
C...        |    *     -
C...        |     *  -
C...        v     - Attachment Point
C... THROAT - - -
C...        ^
C...        |
C...        |
C...    YT  |
C...        |
C...        |
C...
C...  Points 1 and 3 are known
C...  Point 4 is located on the wall
C...  Point 3 is located on the wall
C...  Point 2 is located between points 1 and 3
C...  The C- characteristic goes through points 1 and 3
C...  The C+ characteristic goes through points 2 and 4
C...  The streamline        goes through points 3 and 4
C...
C...                ##
C...   WALL      ##
C...          ## *4
C...       ##   *
C...    ##     *
C... ##       * C+
C...  3*     *
C...    *   *
C...  C- * *
C...     2*
C...       *
C...        * 1
```

```
C
C*********************************************************************
C
C*********************************************************************
C**                                                                 **
C**  INCLUDE STATEMENTS                                             **
C**                                                                 **
C*********************************************************************
C
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'

      DEBUG = .TRUE.

C... convert angles into radians
      TA = AA / RAD   !Attachment angle
      TE = AE / RAD   !Exit angle

      GOTO (10,20) INITIAL
C
C*********************************************************************
C**                                                                 **
C**  INITIAL WALL IS A CIRCLE                                       **
C**                                                                 **
C*********************************************************************
C
 10   CONTINUE

      WRITE(6,*)' '
      ASTR = 'INITIAL WALL PROFILE IS A CIRCLE'
      CALL BANNER(ASTR)
      WRITE(6,*)' '
      WRITE(6,*)'  Where:'
      WRITE(6,*)'         X = RTD * SIN(THETA)'
      WRITE(6,*)'         Y = YT+RTD*(1.-COS(THETA))'
      WRITE(6,*)'  '
      WRITE(6,*)'   and:  THETA = Wall Angle from 0.0 to Attachment Angle'
      WRITE(6,*)'  '
      WRITE(6,FMT='(A,1P2E15.5)')'    Starting Point (X,YT)      = ', 0.0,YT
C
C*********************************************************************
C**                                                                 **
C**  FIND THE COEFFICENTS OF THE SECOND ORDER QUADRATIC WALL        **
C**                                                                 **
C**   CALCULATE THE COEFFICIENTS C1, C2, C3                         **
C**    WHERE Y  = C1 + C2*X + C3*X**2                               **
C**                                                                 **
C*********************************************************************
C
C... From geometry, find the attachment point, the point that ends
C...  the circle and begins the second order quadratic.

      XA = RTD * SIN(TA)
      YA = YT+RTD*(1.-COS(TA))

C... Using the starting and ending angles of the quadratic find
C...  the coefficents

      C3 = (TAN(TE)-TAN(TA))/(2.0*(XE-XA))
      C2 = TAN(TA)-2.0*C3*XA
      C1 = YA-C2*XA-C3*XA**2

      WRITE(6,*)' '
      ASTR = 'SECOND ORDER QUADRATIC WALL CONTOUR, DOWNSTREAM OF CIRCLE'
      CALL BANNER(ASTR)
c     WRITE(6,*)'USING SECOND ORDER QUADRATIC WALL CONTOUR'
c     WRITE(6,*)' ----------------------------------------'
      WRITE(6,*)'  Where:'
      WRITE(6,*)'         Y  = C1 + C2*X + C3*X**2'
      WRITE(6,*)'  '
      WRITE(6,*)'   And:'
      WRITE(6,FMT='(A,1PE15.5)')'         C1 = ',C1
      WRITE(6,FMT='(A,1PE15.5)')'         C2 = ',C2
      WRITE(6,FMT='(A,1PE15.5)')'         C3 = ',C3
      WRITE(6,*)'  '
      WRITE(6,*)'  Starting Location: '
      WRITE(6,FMT='(A,1P2E15.5)')'    Attachment Point (XA,YA)   = ', XA,YA
      WRITE(6,FMT='(A,1P1E15.5)')'    Attachment Angle (Degrees) = ', AA

      YE = C1+C2*XE +C3*XE**2
      LE = C2+ 2.*C3*XE
```

```
      IF(DEBUG)GOTO 30

      RETURN
C
C*********************************************************************
C**                                                                 **
C**  SOLVE FOR THE INTERCEPTION LOCATION OF THE UPWARD CHARACTERISTIC **
C**   AND THE SECOND ORDER QUADRATIC WALL, POINT X4,Y4 AND SLOPE A4   **
C**                                                                 **
C**   LP IS THE TANGENT ANGLE AT POINT X2,Y2                        **
C**                                                                 **
C*********************************************************************
C
 20   CONTINUE

      IF(AA.EQ.AE)THEN
       X4 = (C1-Y2+LP*X2)/(LP-C2)
      ELSE
       X4 = ((LP-C2)-SQRT((LP-C2)**2-4.0*C3*(C1-Y2+LP*X2)))/(2.0*C3)
      ENDIF

      Y4 = C1+C2*X4 +C3*X4**2
      A4 = ATAN(C2+ 2.*C3*X4)

c     IF(DEBUG)THEN

c     WRITE(6,*)' '
c     WRITE(6,*)' LOCATION OF THE INTERSECTION OF THE UPWARD RUNNING',
c    >' CHARACTERISTIC WITH A SECOND ORDER QUADRATIC WALL CONTOUR.'
c     write(6,*)'P( X4 , Y4 ) = ',X4,Y4
c     WRITE(6,*)' '

c     ENDIF


      RETURN
C
C*********************************************************************
C*                                                                  *
C*    MAKE GNU PLOT FOR VIEWING DATA                                *
C*                                                                  *
C*********************************************************************
C
 30   CONTINUE

      WRITE(6,*)' '
      OPEN(UNIT=55,FILE='WALL_DEFINITION.dem',FORM='FORMATTED')
      WRITE(55,500)
 500  FORMAT('# GNUPLOT v3.6 beta multiplot script file',/,
     >'set terminal pdf enhanced color font "Helvetica" fsize 8 size 10.5 in, 8.0 in',/,
     >'set output "WALL_DEFINITION.pdf"',/,
     >'set key left top box',/,
     >'set border',/,
     >'set grid',/,
     >'set style line 1 lt rgb "black"   lw 1 pt 1 ps 1',/,
     >'set style line 2 lt rgb "red"     lw 1 pt 2 ps 1',/,
     >'set style line 3 lt rgb "green"   lw 1 pt 3 ps 1',/,
     >'set style line 4 lt rgb "blue"    lw 1 pt 4 ps 1',/,
     >'set style line 5 lt rgb "orange"  lw 1 pt 5 ps 1',/,
     >'set style line 6 lt rgb "yellow"  lw 1 pt 6 ps 1',/,
     >'show terminal ',/,
     >'set title "WALL DEFINITION"',/,
     >'set xlabel "Axial Length"',/,
     >'set ylabel "Radial Length"',/,
     >'set palette defined (0 0 0 0, 1 0 0 1, 3 0 1 0, 4 1 0 0, 6 1 1 1)',/,
     >'set hidden3d')
      WRITE(55,501)
 501  FORMAT('plot [0:] [0:] "-" using 1:2 with linespoints title "CIRCULAR ARC" ls 1')

      DO Z = 0.0 , TA, TA/25.0
      XZ = RTD * SIN(Z)
      YZ = YT+RTD*(1.-COS(Z))
      WRITE(55,*)XZ,YZ
      ENDDO

      WRITE(55,*)' '
      WRITE(55,*)RTD * SIN(TA), YT+RTD*(1.-COS(TA))
      WRITE(55,*)' '

      DO Z = RTD * SIN(TA), XE , (XE-RTD * SIN(TA))/100.0
```

```
      XZ = Z
      YZ = C1 + C2*XZ + C3*XZ**2
      WRITE(55,*)XZ,YZ
      ENDDO

      WRITE(55,*)' '
      WRITE(55,*)XE,YE

      IF(INITIAL.EQ.2)THEN
      WRITE(55,*)' '
      WRITE(55,*)X2,Y2
      WRITE(55,*)X4,Y4
      ENDIF

      WRITE(55,503)
 503  FORMAT('end')

      CLOSE(55)
      WRITE(6,*)' '
      WRITE(6,*)'  Type: gnuplot WALL_DEFINITION.dem ; open WALL_DEFINITION.pdf'

      END
      SUBROUTINE IVLINE
C
C**********************************************************************
C**                                                                 **
C**   SUBROUTINE IVLINE DETERMINES AN INITIAL VALUE LINE USING      **
C**   SAUERS METHOD. THIS METHOD CONVERTS A MACH 1 START INTO       **
C**   A TRANSONIC START LINE DOWNSTREAM OF THE THROAT.              **
C**                                                                 **
C**********************************************************************
C
C***********************************************************************
C*                                                                    *
C*     TERMINOLOGY FOR SUBROUTINE IVLINE                              *
C*                                                                    *
C* CONTROL VARIABLE:                                                  *
C* ----------------                                                   *
C*                                                                    *
C* DELTA  = '0' FOR PLANER FLOW                                       *
C*        = '1' AXISYMMETRIC FLOW                                     *
C* NI     = ODD NUMBER OF EQUALLY SPACE POINTS ON THE VPRIME = 0 LINE *
C* IUNITS = 1 FOR EE UNITIS                                           *
C*        = 2 FOR SI UNITIS                                           *
C*                                                                    *
C* NOZZLE THROAT GEOMETRY AND PERFORMANCE PARAMETERS                  *
C* -----------------------------------------------                    *
C*                                                                    *
C* YT     = NOZZLE THROAT RADIUS, M (IN)                              *
C* RTU    = NOZZLE THROAT UPSTREAM RADIUS OF CURVATURE, M (IN)        *
C* ALPHA  = COEFFICIENT OF LINEAR AXIAL VELOCITY PERTURBATION, M^-1   *
C* EPS    = LOCATION OF NOZZLE THROAT, M (IN)                         *
C* ASTAR  = CRITICAL SPEED OF SOUND, M/S (FT/SEC)                     *
C* MDOT   = MASS FLOW RATE, KG/S (LBM/SEC)                            *
C* MDOTS  = 1D MASS FLOW RATE, KG/S (LBM/SEC)                         *
C* CD     = DISCHARGE COEFFICIENT MDOT/MDOT,1D                        *
C* F      = THROAT THRUST, N (LBF)                                    *
C* FS     = 1D THROAT THRUST, N (LBF)                                 *
C* LAMBDA = THRUST RATIO, FT/FT,1D                                    *
C*                                                                    *
C***********************************************************************
C*
C*
C***********************************************************************
C*                                                                    *
C*    INCLUDE AND COMMON STATEMENTS                                   *
C*                                                                    *
C***********************************************************************
C*
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C*
C***********************************************************************
C*                                                                    *
C*    FUNCTION STATEMENTS - INPUT IS VELOCITY                         *
C*                                                                    *
C***********************************************************************
C*
      TT(X) = TS - X*X/(2.0*GC*CP)           !static temperature
      AM(X) = X / SQRT(G*GC*RG*TT(X))        !mach number
      PP(X) = PS * (TT(X)/TS)**(G/(G-1.0))  !static pressure
```

```
      RR(X) = PP(X) / RG / TT(X)            !static density
C*
C***********************************************************************
C*                                                                     *
C*    OUTPUT STATEMENTS                                                *
C*                                                                     *
C***********************************************************************
C*
      WRITE(6,*)' '
      ASTR = 'DETERMINE AN INITIAL VALUE LINE: TRANSONIC START LINE'
      CALL BANNER(ASTR)
C*
C***********************************************************************
C*                                                                     *
C*    CHECK TO SEE IF INPUTS ARE VALID FOR SAUERS METHOD               *
C*                                                                     *
C***********************************************************************
C*
      IF(RTU/YT .LE. 1.0)THEN
       WRITE(6,*)' '
       WRITE(6,*)'[31m THE RATIO OF RTU/YT IS INVALID FOR SAUERS METHOD![0m '
       WRITE(6,*)'          RTU/YT = ', RTU/YT
       WRITE(6,*)' IF POSSIBLE, INCREASE THE RATIO TO > 2'
       WRITE(6,*)'  ******** STOPPING THE CODE *********'
       STOP
      ELSEIF(RTU/YT.GT. 1.0 .AND. RTU/YT.LT. 2.0)THEN
       WRITE(6,*)' '
       WRITE(6,*)' WARNING:[33m THE RTU/YT RATIO IS MARGINALLY ACCECPTABLE. [0m'
       WRITE(6,*)'          RTU/YT = ', RTU/YT
       WRITE(6,*)' IF POSSIBLE, INCREASE THE RATIO TO > 2'
      ENDIF
C*
C***********************************************************************
C*                                                                     *
C*    CALCULATE THE REFERENCE PARAMETERS FOR SAUERS METHOD             *
C*                                                                     *
C***********************************************************************
C*
      ALPHA = SQRT((1.0+DELTA)/((G+1.0)*RTU*YT))
      DY    = YT / (NI*1.-1.)
      C1    =-(G+1.0)*ALPHA/(2.0*(3.0+DELTA))
      C2    = (G+1.0)*ALPHA**2/(2.0*(1.0+DELTA))
C*
C***********************************************************************
C*                                                                     *
C*    CALCULATE THE AXIAL DISPLACEMENT DOWNSTREAM OF THE THROAT        *
C*                                                                     *
C***********************************************************************
C*
      EPS   =-(G+1.0)*ALPHA*YT**2/(2.0*(3.0+DELTA))
C*
C***********************************************************************
C*                                                                     *
C*    SET INITIAL VARIABLES                                            *
C*                                                                     *
C***********************************************************************
C*
      IF(DELTA.EQ.1)THEN
       AT    = PI*YT*YT/GL                   !Area at throat
      ELSE
       AT    = YT
      ENDIF

      Y(NI*2) = 0.0                          !inital radial location
      MDOT  = 0.0                           !inital mass flow
      F     = 0.0                           !inital thrust
      AS    = SQRT(2.0*G*GC*RG*TS/(G+1.0))  !speed of sound
      MDOTS = AS*RR(AS)*AT                  !mass
      FS    = PP(AS)*AT+MDOTS*AS/GC         !force
      FOD   = FS-PA*AT                      !ambient force
C*
C***********************************************************************
C*                                                                     *
C*    CALCULATE LOCATION AND PROPERTIES OF INITIAL VALUE LINE POINTS   *
C*                                                                     *
C***********************************************************************
C*
      WRITE(6,1000)
 1000 FORMAT('    POINT  ','   X BAR    ', '      Y      ',' U VELOCITY '
     >, ' V VELOCITY ','MACH NUMBER ', 'VELOCITY MAG',' FLOW ANGLE ',
     >' PRESSURE   ','  DENSITY   ','TEMPERATURE ',' MASS FLOW  ')
```

```
      DO 10 I = 1, NI

c      J = I + 100
       J = I + NI*2.0 - 1
C*
C*********************************************************************
C*                                                                   *
C*    CALCULATE Y SPACING                                            *
C*                                                                   *
C*********************************************************************
C*
       IF(ST.EQ.0.0)THEN
        IF(I.GT.1) Y(J)=Y(J-1)+DY               !Evenly space y
       ELSE
        ZETA = (I*1.-1.)/(NI*1.-1.)
        IF(I.GT.1) Y(J)=YT*(1.-(1.-ZETA)**2)   !Quaddratic spacing for y
       ENDIF
C*
C*********************************************************************
C*                                                                   *
C*    FIND AXIAL LOCATION AND VELOCITY USING SAUERS METHOD           *
C*                                                                   *
C*********************************************************************
C*
       X(J) = C1 * Y(J)**2 + ERROR
       U(J) = AS * (1. + ALPHA*X(J)+C2*Y(J)**2)

C... Off set x term
       X(J) = X(J) - EPS

       T    = TT(U(J))
       M    = AM(U(J))
       PZ   = PP(U(J))
       RZ   = RR(U(J))

       CI   = 3.0+(-1.0)**I
       IF((I.EQ.1).OR.(I.EQ.NI)) CI = 1.0
C*
C*********************************************************************
C*                                                                   *
C*    FIND AREA - Correct for planar/axisymmetric flow               *
C*       DELTA = 0 for planar, 1 for axisymmetric                    *
C*                                                                   *
C*********************************************************************
C*
       IF(I.NE.1)THEN
        AREA = PI * (Y(J)**2 - Y(J-1)**2) * DELTA + (1-DELTA)*(Y(J)-Y(J-1))
       ELSE
        AREA = 0.0
       ENDIF
C*
C*********************************************************************
C*                                                                   *
C*    FIND MASS FLOW AND FORCE                                       *
C*                                                                   *
C*********************************************************************
C*
       MDOT = MDOT + RZ*U(J)*AREA
       F    = F +((PZ-PA)+RZ*U(J)**2/GC)*AREA

       V(J) = 0.0
       Q(J) = U(J)        !VELOCITY MAG
       A(J) = 0.0         !FLOW ANGLE
       P(J) = PZ
       R(J) = RZ
       PZ   = PZ/GL
C*
C*********************************************************************
C*                                                                   *
C*    OUTPUT THE FLOW VARIABLES                                      *
C*                                                                   *
C*********************************************************************
C*
       JJ = NI-I+1 ! axial index

       WRITE(6,2001)I,JJ,X(J),Y(J),U(J),V(J),M,Q(J),A(J),PZ,RZ,T,MDOT
       WRITE(13,2001)I,JJ,X(J),Y(J),U(J),V(J),M,Q(J),A(J),PZ,RZ,T,MDOT
C*
C*********************************************************************
C*                                                                   *
```

```
C*    PLACE THE FLOW VARIABLES INTO STORAGE                                 *
C*                                                                          *
C***************************************************************************
C*
       XY(I,JJ,1)    = X(J)
       XY(I,JJ,2)    = Y(J)
       FLOW(I,JJ,1) = U(J)
       FLOW(I,JJ,2) = V(J)
       FLOW(I,JJ,3) = M
       FLOW(I,JJ,4) = Q(J)
       FLOW(I,JJ,5) = A(J)
       FLOW(I,JJ,6) = PZ
       FLOW(I,JJ,7) = RZ
       FLOW(I,JJ,8) = T
C*
C***************************************************************************
C*                                                                          *
C*    END OF I DO LOOP                                                      *
C*                                                                          *
C***************************************************************************
C*
 10   ENDDO

C*
C***************************************************************************
C*                                                                          *
C*    OUTPUT THE MASS FLOW AND THRUST                                       *
C*                                                                          *
C***************************************************************************
C*
c     IF(ST.EQ.0.0)THEN
c      MDOT = MDOT*2.0*PI*DY/(3.0*GL)
c      F    = F*2.0*PI*DY/(3.0*GL)
c     ENDIF

      CD   = MDOT/MDOTS            !Nozzle Discharge Coefficient
      ETAF = F/FOD                 !Thrust ratio (efficiency)
      ETAI = ETAF/CD               !Thrust ratio / Nozzle Discharge

      WRITE(6,*)' '
      WRITE(6,2002)MDOT,MDOTS,CD,F,FOD,ETAF,ETAI
C
C***************************************************************************
C*                                                                          *
C*    MAKE GNU PLOT FOR VIEWING DATA                                        *
C*                                                                          *
C***************************************************************************
C
      IF(DEBUG)THEN

      WRITE(6,*)' '
      OPEN(UNIT=55,FILE='STARTING_CONDITIONS.dem',FORM='FORMATTED')
      WRITE(55,500)
 500  FORMAT('# GNUPLOT v3.6 beta multiplot script file',/,
     >'set terminal pdf enhanced color font "Helvetica" fsize 8 size 10.5 in, 8.0 in',/,
     >'set output "STARTING_CONDITIONS.pdf"',/,
     >'set key left top box',/,
     >'set border',/,
     >'set grid',/,
     >'set style line 1 lt rgb "black"   lw 1 pt 1 ps 1',/,
     >'set style line 2 lt rgb "red"     lw 1 pt 2 ps 1',/,
     >'set style line 3 lt rgb "green"   lw 1 pt 3 ps 1',/,
     >'set style line 4 lt rgb "blue"    lw 1 pt 4 ps 1',/,
     >'set style line 5 lt rgb "orange"  lw 1 pt 5 ps 1',/,
     >'set style line 6 lt rgb "yellow"  lw 1 pt 6 ps 1',/,
     >'show terminal ',/,
     >'#set title "STARTING NOZZLE FLOW CONDITIONS"',/,
     >'set ylabel "Radial Length"',/,
     >'set autoscale x',/,
     >'set autoscale y',/,
     >'set multiplot layout 2, 1 title "STARTING NOZZLE FLOW CONDITIONS"',/,
     >'set size 0.5,0.5',/,
     >'set origin 0,0',/,'set xlabel "Density (kg/m^3)"',/,
     >'plot [:] [:] "fort.7" using 11:4 with lines notitle ls 1',/,
     >'set size 0.5,0.5',/,
     >'set origin 0,0.49',/,'set xlabel "Pressure (Pa)"',/,
     >'plot [:] [:] "fort.7" using 10:4 with lines notitle ls 1',/,
     >'set size 0.5,0.5',/,
     >'set origin 0.5,0.49',/,'set xlabel "Temperture (K)"',/,
     >'plot [:] [:] "fort.7" using 12:4 with lines notitle ls 1',/,
     >'set size 0.5,0.5',/,
```

```
     >'set origin 0.5,0.0',/,'set xlabel "MACH NUMBER"',/,
     >'plot [:] [:] "fort.7" using 7:4 with lines notitle ls 1')

      CLOSE(55)

      WRITE(6,*)' '
      WRITE(6,*)'  Type: gnuplot STARTING_CONDITIONS.dem ; open STARTING_CONDITIONS.pdf'

      ENDIF
C*
C**********************************************************************
C*                                                                    *
C*    RETURN TO MAIN PROGRAM                                          *
C*                                                                    *
C**********************************************************************
C*
      RETURN
C*
C**********************************************************************
C*                                                                    *
C*    FORMAT STATEMENTS                                               *
C*                                                                    *
C**********************************************************************
C*
 2001 FORMAT(2I5,1P11E12.4)
 2002 FORMAT(/,1P,
     >       9X,'Mass Flow Rate      = ',E12.3,/,
     >       9X,'Mass Flow Rate (1D) = ',E12.3,/,
     >       9X,'Nozzle Discharge Coefficient = ',E12.3,/,
     >       /,
     >       9X,'Thrust                       = ',E12.3,/,
     >       9X,'Thrust (1D)                  = ',E12.3,/,
     >       9X,'Thrust ratio (efficiency) = ',E12.5,/,
     >       9X,'Thrust ratio / Nozzle Discharge = ',E12.5)
       END

      SUBROUTINE TURNING
C
C**********************************************************************
C*                                                                    *
C*    THIS ROUTINE COMPUTE THE TURNING REGION FLOWFIELD               *
C*                                                                    *
C**********************************************************************
C

C...
C...  This subroutine computes the turning region.
C...
C...  Points 1, 2 and 3 are known
C...  Point 4 is the intersection of points 1 and 2
C...  Point 5 is located between points 1 and 3
C...  The C- characteristic goes through points 2 and 4
C...  The C+ characteristic goes through points 1 and 4
C...  The streamline        goes through points 5 and 4
C...
C...
C...  3*
C...    *               * Final Characteristic Line
C...     *      4      *
C...    5 *******     *
C...       *   * *   *
C...        * *   * *
C...       1 *     * 2
C...          *   *
C...           * *
C...            *
C...
C...
C... Calculate the solution at an interior point
C...
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C*
C**********************************************************************
C*                                                                    *
C*    SET CONSTANTS                                                   *
C*                                                                    *
C**********************************************************************
C*
      ITER = 0
C*
```

```
C**********************************************************************
C*                                                                    *
C*    CALCULATE THE SLOPE BETWEEN POINTS P1 AND P3                    *
C*                                                                    *
C**********************************************************************
C*
      IF(X1.EQ.X3)THEN
      L13 = 0.0
      ELSE
      L13 = (Y1-Y3)/(X1-X3)
      ENDIF
c     write(6,*)'L13 = ',L13
C*
C**********************************************************************
C*                                                                    *
C*    CALCULATE THE PROPORTIES AT POINT P1 FOR THE CHARA +            *
C*                                                                    *
C**********************************************************************
C*
      CALL THERMO (Q1,P1,R1,T,C,M)
      LP = TAN(A1+ASIN(1./M))
c     write(6,*)'ERROR = ',ERROR
c     write(6,*)'LP = ',LP
      QP = GC*SQRT(M**2-1.)/(R1*Q1**2)
      SP = DELTA/(M*COS(A1+ASIN(1./M)))

      IF(Y2.EQ.0.0) SP = SP * SIN(A2)/Y2
      IF(Y2.GT.0.0) SP = SP * SIN(A1)/Y1
C*
C**********************************************************************
C*                                                                    *
C*    CALCULATE THE PROPORTIES AT POINT P2 FOR THE CHARA -            *
C*                                                                    *
C**********************************************************************
C*
      CALL THERMO (Q2,P2,R2,T,C,M)

      LM = TAN(A2-ASIN(1./M))
c     WRITE(6,*)'LM = ',LM
      QM = GC*SQRT(M**2-1.)/(R2*Q2**2)
      SM = DELTA*SIN(A2)/(Y2*M*COS(A2-ASIN(1./M)))
C*
C**********************************************************************
C*                                                                    *
C*    GUESS THE STREAMLINE ANGLE FOR P5                               *
C*                                                                    *
C**********************************************************************
C*
      A5 = 0.5*(A1+A3)
      A4 = A5
C*
C**********************************************************************
C*                                                                    *
C*    PROJECT CHARACTERSITIC LINE AND FIND UNKNOWN POINT P4            *
C*                                                                    *
C**********************************************************************
C*
 10   X4 = (Y1-Y2-LP*X1+LM*X2)/(LM-LP)
      Y4 = Y1+LP*(X4-X1)
c     WRITE(6,*)'X4,Y4 = ',X4,Y4

      IF(Y4.LT.0.0)RETURN

      TP = -SP*(X4-X1)+QP*P1+A1
      TM = -SM*(X4-X2)+QM*P2-A2

      K  = 1
C*
C**********************************************************************
C*                                                                    *
C*    INTERPOLATE BACK FROM POINT 4 ALONG THE STREAM LINE TO FIND     *
C*     POINT 3 (THE POINT BETWEEN 1 AND 2), ANGLE OF STREAMLINE       *
C*                                                                    *
C**********************************************************************
C*
 20   L0 = TAN(0.5*(A5+A4))
C*
C**********************************************************************
C*                                                                    *
C*    CALCULATE POINT P5 PROPERTIES                                   *
C*                                                                    *
```

```
C*********************************************************************
C*
      IF(X1.NE.X3)THEN
C...intercept
       X5 = (Y4-Y1-L0*X4+L13*X1)/(L13-L0)
       Y5 = Y4+L0*(X5-X4)
c      WRITE(6,*)'X5,Y5= ',X5,Y5
c... location ratio between points 1 and 3
       D  = (Y5-Y1)/(Y3-Y1)
c... linear interp for angle 5
       A5 = A1+D*(A3-A1)

      ELSE  !on wall

       X5 = X1
       Y5 = Y1
       D  = 0.0
       A5 = A1

      ENDIF

      IF(ITER.EQ.0)A4=A5
      IF(K.GT.1 .AND. ABS(Y5-YC).LT.ERROR) GOTO 30
C...ERROR in decode
      IF(K.GT.101)THEN
       WRITE(6,*)' ERROR: CAN NOT DECODE FLOW ANGLE!'
       WRITE(6,*)'Y5 =',Y5
       WRITE(6,*)'YC =',YC
       WRITE(6,*)'Y5-YC =',Y5-YC
       WRITE(6,*)' '
       WRITE(6,*)'X1,Y1 = ',X1,Y1
       WRITE(6,*)'X2,Y2 = ',X2,Y2
       WRITE(6,*)'X3,Y3 = ',X3,Y3
       WRITE(6,*)'X4,Y4 = ',X4,Y4

       STOP
      ENDIF

      XC = X5
      YC = Y5

      K  = K + 1
      GOTO 20

 30   Q5 = Q1+D*(Q3-Q1)
      P5 = P1+D*(P3-P1)
      R5 = R1+D*(R3-R1)
C*
C*********************************************************************
C*                                                                   *
C*    CALCULATE POINT P4 PROPERTIES                                  *
C*                                                                   *
C*********************************************************************
C*
      IF(ITER.GT.0)GOTO 40

      Q4 = Q5
      P4 = P5
      R4 = R5

 40   P0 = 0.5*(P5+P4)
      R8 = 0.5*(R5+R4)
      Q0 = 0.5*(Q5+Q4)
      R0 = R8*Q0/GC

      CALL THERMO (Q0,P0,R8,T,C,M)

      A0 = C**2/GC
      T01=R0*Q5+P5
      T02=P5-A0*R5

      P4 = (TP+TM)/(QP+QM)
      A4 = TP-QP*P4
      Q4 = (T01-P4)/R0
      R4 = (P4-T02)/A0
C*
C*********************************************************************
C*                                                                   *
C*    TEST FOR CONVERGENCE                                           *
C*                                                                   *
C*********************************************************************
```

```
C*
c     WRITE(6,*)' '
c     WRITE(6,100)' R0                   = ',R0
c     WRITE(6,100)' A0                   = ',A0
c     WRITE(6,100)' T01                  = ',T01
c     WRITE(6,100)' T02                  = ',T02
c     WRITE(6,100)' Q+                   = ',QP
c     WRITE(6,100)' S+                   = ',SP
c     WRITE(6,100)' T+                   = ',TP
c     WRITE(6,100)' Q-                   = ',QM
c     WRITE(6,100)' S-                   = ',SM
c     WRITE(6,100)' T-                   = ',TM

C... First time thru, skip the convergence test
      IF(ITER.EQ.0)    GOTO 50

C... Reached the number of times on the correction
      IF(ITER.EQ.ICOR)THEN
       WRITE(6,*)' '
       WRITE(6,*)' Stopping the code:',
     >           ' Reached the ICOR user set limit!'
       WRITE(6,*)' ABS(X4-XD),E1',ABS(X4-XD),E1
       WRITE(6,*)' ABS(Y4-YD),E1',ABS(Y4-YD),E1
       WRITE(6,*)' ABS(P4-PD),E2*PD',ABS(P4-PD),E2*PD
       WRITE(6,*)' ABS(R4-RD),E3*RD',ABS(R4-RD),E3*RD
       WRITE(6,*)' ABS(Q4-QD),E4*QD',ABS(Q4-QD),E4*QD
       WRITE(6,*)' ABS(A4-AD),E5*AD',ABS(A4-AD),E5*AD
       STOP
      ENDIF

      IF((ABS(X4-XD).GT.E1)    .OR.  (ABS(Y4-YD).GT.E1))    GOTO 50
      IF((ABS(P4-PD).GT.E2*PD) .OR.  (ABS(R4-RD).GT.E3*RD)) GOTO 50

      IF((ABS(Q4-QD).LT.E4*QD) .AND. (ABS(A4-AD).LT.E5*AD)) RETURN
C*
C***********************************************************************
C*                                                                     *
C*    CALCULATE THE COEFFICIENTS FOR THE CORRECTOR                     *
C*                                                                     *
C***********************************************************************
C*
 50   ITER = ITER + 1

      XD = X4
      YD = Y4
      PD = P4
      RD = R4
      QD = Q4
      AD = A4

      P0 = 0.5*(P1+P4)
      R8 = 0.5*(R1+R4)
      Q0 = 0.5*(Q1+Q4)
      A0 = 0.5*(A1+A4)
      Y0 = 0.5*(Y1+Y4)

      CALL THERMO (Q0,P0,R8,T,C,M)

      LP = TAN(A0+ASIN(1.0/M))
      QP = GC*SQRT(M**2-1.0)/(R8*Q0**2)

      P0 = 0.5*(P1+P4)
      R8 = 0.5*(R1+R4)

      SP = DELTA*SIN(A0)/(Y0*M*COS(A0+ASIN(1.0/M)))

      Q0 = 0.5*(Q2+Q4)
      A0 = 0.5*(A2+A4)
      Y0 = 0.5*(Y2+Y4)

      CALL THERMO (Q0,P0,R8,T,C,M)

      LM = TAN(A0-ASIN(1.0/M))
      QM = GC*SQRT(M**2-1.0)/(R8*Q0**2)

      SM = DELTA*SIN(A0)/(Y0*M*COS(A0-ASIN(1.0/M)))

      GOTO 10                               !RECALCULATE POINTS P4 AND P5
C
C***********************************************************************
C*                                                                     *
```

```
C*    FORMAT STATEMENTS                                                 *
C*                                                                      *
C***********************************************************************
C
 100  FORMAT(A,1P2E15.5)
C*
C***********************************************************************
C*                                                                      *
C*    END OF LINE                                                       *
C*                                                                      *
C***********************************************************************
C*
      END
      SUBROUTINE CIRCULAR_FLOWFIELD
C
C***********************************************************************
C*                                                                      *
C*    CALCULATE THE FLOW FIELD FROM THE CIRCULAR ARC THROAT CONTOUR    *
C*                                                                      *
C***********************************************************************
C
C
C***********************************************************************
C*                                                                      *
C*     TERMINOLOGY FOR SUPERSONIC FLOW METHOD OF CHARACTERISTICS       *
C*                                                                      *
C* CONTROL VARIABLE:                                                    *
C* ----------------                                                     *
C*                                                                      *
C* DELTA  = '0' FOR PLANER FLOW                                         *
C*        = '1' AXISYMMETRIC FLOW                                       *
C* ICOR   = NUMBER OF APPLICATIONS OF THE CORRECTOR DESIRED             *
C* E1     = CONVERGENCE TOLERANCE FOR LOCATION, M (IN)                  *
C* E2     = CONVERGENCE TOLERANCE FOR VELOCITY, M/S (FT/SEC)            *
C* GC     = 1.0 M-KG/N-S^2 OR 32.174 FT-LBM/LBF-S^2                     *
C* GL     = 1.0 M^2/M^2 OR 144.0 IN^2/FT^2                              *
C* ST     = 0.0 EVEN SPACE STARTLINE, 1.0 QUADRATIC SPACED              *
C*                                                                      *
C* GAS THERMODYNAMIC PROPERTIES & STAGNATION PROPERTIES:                *
C* ---------------------------------------------------                  *
C*                                                                      *
C* G      = RATIO OF SPECIFIC HEATS                                     *
C* RG     = GAS CONSTANT, J/KG-K (FT-LBF/LBM-R)                         *
C* TS     = STAGNATION TEMPERATURE, K (R)                               *
C* PS     = STAGNATION PRESSURE, N/M^2 (LBF/IN^2)                       *
C* PA     = AMBIENT PRESSURE, N/M^2 (LBF/IN^2)                          *
C*                                                                      *
C* FLOW FIELD PROPERTIES:                                               *
C* ---------------------                                                *
C*                                                                      *
C* X      = AXIAL COORDINATE, M (IN)                                    *
C* Y      = RADIAL COORDINATE, M (IN)                                   *
C* U      = AXIAL VELOCITY, M/S (FT/S)                                  *
C* V      = RADIAL VELOCITY, M/S (FT/S)                                 *
C* Q      = VELOCITY MAGNITUDE, M/W (FT/S)                              *
C* A      = FLOW ANGLE, RAD                                             *
C* P      = STATIC PRESSURE, N/M^2 (LBF/IN^2)                           *
C* R      = STATIC DENSITY, KG/M^3 (LBM/FT^3)                           *
C* T      = STATIC TEMPERATURE, K (R)                                   *
C* C      = SPEED OF SOUND, M/S (FT/S)                                  *
C* M      = MACH NUMBER                                                 *
C* EMD    = DESIGN MACH NUMBER                                          *
C* 1,2,3, = DENOTES PROPERTIES AT POINTS                                *
C*                                                                      *
C* TERMINOLOGY EMPLOYED:                                                *
C* --------------------                                                 *
C*                                                                      *
C* L      = TAN(THETA+-ALPHA)                                           *
C* Q      = (U^2-C^2), M^2/S^2 (FT^2/S^2)                               *
C* R      = 2UV-L(U^2-C^2) M^2/S^2 (FT^2/S^2)                           *
C* S      = DELTA*C^2*V/Y, M^2/S^3 (FT^2/SEC^3-IN)                      *
C* T      = S*DEL(X)+Q*U+R*V, M^3/S^3 (FT^3/S^3)                        *
C* +/-    = DENOTES + OR - CHARACTERISTIC CURVE                         *
C*                                                                      *
C***********************************************************************
C
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C
C**********************************************************************
C**                                                                   **
```

```
C**   Function Statements                                              **
C**                                                                    **
C***********************************************************************
C
      TM(B) = B / TZERO
      ZMUIFD(B) = ZMZERO*TM(B)**1.5*((1.+CONSU)/(TM(B)+CONSU))
      RE(B) = R(1) * Q(1) * X(1) / ZMUIFD(B)
      BL(B) = RE(B)**(1./8.)
C
C************************************************************************
C*                                                                      *
C*    OUTPUT THE HEADER                                                 *
C*                                                                      *
C************************************************************************
C
      WRITE(6,*)' '
      ASTR = 'CALCULATE THE FLOW FIELD FROM THE CIRCULAR ARC THROAT CONTOUR'
      CALL BANNER(ASTR)
C
C************************************************************************
C*                                                                      *
C*    INITIALIZE THE CONSTANTS                                          *
C*                                                                      *
C************************************************************************
C
      II   = NI + 1          !starting i index
      IK   = II + NT - 1     !ending i index
      IMAX = IK

      J2   = 2*NI-1          !this is true from here on out, reanforce it.

      L    = 0
      III  = NI + NT
      IEND = 0
      KW = 1
C
C************************************************************************
C*                                                                      *
C*    SET THE STOPING POINT FOR THE CALCULATION WITH THE CIRCULAR ARC  *
C*    WALL.                                                             *
C*                                                                      *
C************************************************************************
C
      TA = AA / RAD
      XA = RTD * SIN(TA)
      YA = YT+RTD*(1.-COS(TA))
      WRITE(6,*)' '
      WRITE(6,*)' STOPPING POINT (XA,YA)', XA,YA
      WRITE(6,*)' '
C
C************************************************************************
C*                                                                      *
C*    BEGIAN THE LOOPING OVER EACH MARCHING I INDEX                     *
C*                                                                      *
C************************************************************************
C
      DO I = II , IK

       N    = 0
       MFRT = 0
       WRITE(8,*)' '
       WRITE(6,*)' '
       CALL BANNER(ASTR)
       WRITE(6,1010)
C
C************************************************************************
C*                                                                      *
C*    FIND THE STARTING WALL POINT TO BEGIN THE DOWNWARD CALCULATION   *
C*                                                                      *
C************************************************************************
C
       DA  = AA/(NT*1.*RAD)
       A4 = DA*(I*1.-NI*1.)               !FLOW AND WALL ANGLE
       X4 = RTD*SIN(A4)
       Y4 = YT + RTD*(1.0-COS(A4))
       L0 = -X4/(Y4-(RTD+YT))
c      WRITE(6,*)' DA = ',DA*RAD
c      WRITE(6,*)' (X4,Y4) = ',X4,Y4
c      WRITE(6,*)' L0 = ',L0
C
C************************************************************************
```

```
C*                                                                    *
C*    IF THIS POINT IS PAST THE ATTACHMENT POINT THEN RESET TO THE    *
C*     ATTCHMENT POINT AND CONTINUE                                   *
C*                                                                    *
C**********************************************************************
C
       IF(X4.GT.XA .OR. Y4.GT.YA)THEN
        X4 = XA
        Y4 = YA
       ENDIF

 91   CONTINUE
C
C**********************************************************************
C*                                                                    *
C*    BEGIN MARCHING FROM THE WALL TO THE CENTERLINE, J INDEX         *
C*                                                                    *
C**********************************************************************
C
      DO 160  J = 1 , J2+1

       J3 = J

       IF(J.EQ.1)THEN           !On the wall
        J2 = J2 + 1
        N  = N  + 1
        CALL MOVE(3,N)          !Move the first point into P3
        CALL MOVE(1,N+1)        !Move the second point into P1
        CALL INVERSE_WALL_POINT

        IF(X2.GT.X1)THEN
c        L  = L + 1
         J2 = J2- 2
         GOTO 91
        ENDIF

       ELSEIF(J.EQ.J2)THEN    !on centerline
        CALL MOVE(1,J-1)      !move the upper point into point 1
        CALL MOVE(6,0)        !move the old P2 point from INTERIOR_POINT into P3
        CALL AXIS

       ELSE                   !on interior point
        CALL MOVE(1,J-1)      !move the upper point into point 1
c       CALL MOVE(2,J)        !move the current point into point 2
        CALL MOVE(2,J+N-1)    !move the current point into point 2
        CALL INTERIOR_POINT

       ENDIF

       CALL MOVE(5,J)         !save point 4 into current array
C
C**********************************************************************
C*                                                                    *
C*    CALCULATE THE MASS FLOW RATE                                    *
C*                                                                    *
C**********************************************************************
C
       IF(J.EQ.1)THEN
        AREA_X = 0.0
        AREA_Y = 0.0
        DEL_X  = 0.0
        DEL_Y  = 0.0
        R_AVE  = 0.0
        U_AVE  = 0.0
        V_AVE  = 0.0
       ELSE
        DEL_X  = ABS(X4-X(J-1))
        DEL_Y  = ABS((Y4**2 - Y(J-1)**2 ))*DELTA + (1-DELTA)*ABS(Y4-Y(J-1))
        AREA_X = PI * DEL_Y * DELTA + (1-DELTA)*DEL_Y
        AREA_Y = 2.0 * PI * (Y4 + Y(J-1))/2.0*DEL_X*DELTA + (1-DELTA)*DEL_X
        R_AVE  = (R4 + R(J-1))/2.0
        U_AVE  = (U4 + U(J-1))/2.0
        V_AVE  = (V4 + V(J-1))/2.0
       ENDIF

        MFR     = R_AVE*(U_AVE*AREA_X+V_AVE*AREA_Y)
        MFRT    = MFR + MFRT
        ERROR_M = (MFRT-MDOT)/MDOT

       AP = A4*RAD
       PP = P4/GL
```

```
      LJ = L + J
C*
C***********************************************************************
C*                                                                     *
C*    OUTPUT THE FLOW VARIABLES                                        *
C*                                                                     *
C***********************************************************************
C*
       IF (KW.EQ.1)THEN
        WRITE(6,1020)I,LJ,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
        WRITE(8,1020)I,LJ,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
        IF(J.EQ.1)WRITE(3,1020)I,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4

IF(J.EQ.1)WRITE(32,1020)I,J,X4,Y4,TM(T4),ZMUIFD(T4),RE(T4),BL(T4),Y4+BL_SCALE*0.1404*((1.+X4)**0.
125-1.0)
        IF(J.EQ.J2+1)WRITE(4,1020)I,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
        IF(J.EQ.J2)write(6,*)' MASS FLUX = ',MFRT,
     >   ' PERCENT ERROR = ',ERROR_M*100.

       ENDIF
C*
C***********************************************************************
C*                                                                     *
C*    PLACE THE FLOW VARIABLES INTO STORAGE                            *
C*                                                                     *
C***********************************************************************
C*
         XY(I,J,1)   = X4
         XY(I,J,2)   = Y4
         IF(J.GT.1)THEN
          IF( Y4.GE.XY(I,J-1,2))THEN
           WRITE(6,*)'Error in Y!', J,J-1
          ENDIF
         ENDIF
         FLOW(I,J,1) = U4
         FLOW(I,J,2) = V4
         FLOW(I,J,3) = M4
         FLOW(I,J,4) = Q4
         FLOW(I,J,5) = A4
         FLOW(I,J,6) = P4
         FLOW(I,J,7) = R4
         FLOW(I,J,8) = T4
c        JMAX = MAX(J,JMAX)

c      IF(X4.GT.XE) GOTO 170
       IF(X4.GT.XE)THEN
        EXIT_END=.TRUE.
        RETURN
       ENDIF
C*
C***********************************************************************
C*                                                                     *
C*    END OF J INDEX LOOP                                              *
C*                                                                     *
C***********************************************************************
C*
 160  CONTINUE
C*
C***********************************************************************
C*                                                                     *
C*    AT THE END OF EACH J INDEX LOOP WE NEED TO CHECK IF THE          *
C*     CENTERLINE MACH NUMBER HAS REACHED THE DESIGNED MACH NUMBER.    *
C*                                                                     *
C***********************************************************************
C*
      IF(NOZ.EQ.0 .AND. Y4.EQ.0.0 .AND. M4.GE.EMD)THEN

       J = J - 1
       J_START  =  1
       J_END    =  J
       L_OFFSET =  L
       I_LINE   =  I

C... need to backspace the plot files

       BACKSPACE(3)
       write(6,*)'I_LINE,J_END =',I_LINE,J_END

       DO LL = 1 , J
        BACKSPACE(8)
       ENDDO
```

```
       DO LL = 1 , J
        WRITE(8,*)' '
       ENDDO
C*
C***********************************************************************
C*                                                                     *
C*    FIND THE LOCATION WHERE MACH NUMBER = DESIGNED MACH NUMBER       *
C*                                                                     *
C***********************************************************************
C*
       CALL MLINE(J_START,J_END,I_LINE,L_OFFSET,8)

       IMAX = I
       J =J +1

       DESIGNED_M=.TRUE.

       RETURN

      ENDIF
C
C***********************************************************************
C*                                                                     *
C*    MAKE SOME CORRECTIONS                                            *
C*                                                                     *
C***********************************************************************
C
      IF(IEND.EQ.1) J2= J2- 1
      GOTO 180

c170  CONTINUE
c     J2= J - 1
c     IEND = 1

 180  CONTINUE
C
C***********************************************************************
C*                                                                     *
C*    SMOOTH THE PROFILE                                               *
C*                                                                     *
C***********************************************************************
C
      IF(SMOOTH)THEN
       DO LL = 1 , J
        BACKSPACE(8)
       ENDDO
       J_START  =  1
       J_END    =  J2
       L_OFFSET =  L
       I_LINE   =  I
       I_FILE   =  8
       CALL SMOOTH_PROFILE(J_START,J_END,I_LINE,L_OFFSET,I_FILE)
      ENDIF
C
C***********************************************************************
C*                                                                     *
C*    END OF I INDEX LOOP                                              *
C*                                                                     *
C***********************************************************************
C
      ENDDO
C
C***********************************************************************
C*                                                                     *
C*    SMOOTH THE PROFILE                                               *
C*                                                                     *
C***********************************************************************
C
c     IF(SMOOTH)THEN
c      DO LL = 1 , J
c       BACKSPACE(8)
c      ENDDO
c      J_START  =  1
c      J_END    =  J2
c      L_OFFSET =  L
c      I_LINE   =  IK
c      I_FILE   =  8
c      CALL SMOOTH_PROFILE(J_START,J_END,I_LINE,L_OFFSET,I_FILE)
c     ENDIF
```

```
c     WRITE(3,1010)
      WRITE(4,1010)
C
C***********************************************************************
C*                                                                    *
C*    FORMAT STATEMENTS                                               *
C*                                                                    *
C***********************************************************************
C
c1010 FORMAT(5X,'I',4X,'J',6X,'X',9X'Y',9X,'U',9X,'V',9X,'M',9X,'Q',
c    >9X,'A',9X,'P',12X,'R',12X,'T',/)
 1010 FORMAT(/,'    I    J ','   X         ', '      Y       ',' U VELOCITY ', ' V VELOCITY ',
     >'MACH NUMBER ', 'VELOCITY MAG',' FLOW ANGLE ',' PRESSURE   ','  DENSITY   ',' TEMPERATURE')
 1020 FORMAT(2I5,1P10E12.4)
      RETURN
C
C***********************************************************************
C*                                                                    *
C*    END OF LINE                                                     *
C*                                                                    *
C***********************************************************************
C
      END
      SUBROUTINE IVSTART
C
C**********************************************************************
C**                                                                  **
C**   SUBROUTINE IVSTART DETERMINES AN INITIAL VALUE LINE USING      **
C**   USER DEFINED INPUT START LINE.                                 **
C**                                                                  **
C**   *****PROFILE MUST START AT CENTERLINE AND MOVE UP TO WALL****  **
C**                                                                  **
C**********************************************************************
C*                                                                    *
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'

      DIMENSION X8(NX),Y8(NX),U8(NX),V8(NX),P8(NX),RHO8(NX)
      DIMENSION X9(NX),Y9(NX),U9(NX),V9(NX),P9(NX),RHO9(NX),A9(NX),Q9(NX)
      DIMENSION VT(NX),ANGLE(NX)

C*
C***********************************************************************
C*                                                                    *
C*    FUNCTION STATEMENTS - INPUT IS VELOCITY                         *
C*                                                                    *
C***********************************************************************
C*
      TT(X) = TS - X*X/(2.0*GC*CP)           !static temperature
      AM(X) = X / SQRT(G*GC*RG*TT(X))        !mach number
      PP(X) = PS * (TT(X)/TS)**(G/(G-1.0))   !static pressure
      RR(X) = PP(X) / RG / TT(X)             !static density
C*
C***********************************************************************
C*                                                                    *
C*    SET CONSTANTS                                                   *
C*                                                                    *
C***********************************************************************
C*
      DEBUG = .TRUE.
      I          = 0
      MDOT       = 0.0
      PI         = 4.0 * ATAN(1.0)
      RAD        = 180./4.0/ATAN(1.0)
      NUM_POINTS = NI
      YY         = 0.0
C*
C***********************************************************************
C*                                                                    *
C*    OUTPUT STATEMENTS                                               *
C*                                                                    *
C***********************************************************************
C*
      WRITE(6,*)' '
      ASTR = 'READ IN THE INITIAL VALUE LINE FROM FILE: Profile.dat'
      CALL BANNER(ASTR)
C*
C***********************************************************************
C*                                                                    *
C*    OPEN INPUT FILE                                                 *
```

```
C*                                                                    *
C**********************************************************************
C*
      INQUIRE(FILE='Profile.dat',EXIST=EXIST)

      IF (.NOT.EXIST) THEN
       WRITE(*,13) FILEN
       STOP
      ENDIF

      OPEN(UNIT=2,FILE='Profile.dat',form='formatted')
C*
C**********************************************************************
C*                                                                    *
C*    READ IN THE INPUT FILE                                          *
C*                                                                    *
C**********************************************************************
C*
      IC = 0

      DO N = 1 ,10000
       read(2,*,end=100)i,j,k,X8(N),Y8(N),Z8,rho8(N),U8(N),V8(N),W8,P8(N)
       IF(N.EQ.1)Y8(1) = 0.0
       IF(N.EQ.1)V8(1) = 0.0
c      write(6,200)X8(N),Y8(N),Z8,rho8(N),U8(N),V8(N),W8,P8(N)
       IC = IC + 1
      ENDDO

 100  continue

      CLOSE(2)

      DY    = Y8(IC) / (NUM_POINTS*1.-1.)    !Spacing
C*
C**********************************************************************
C*                                                                    *
C*    SMOOTH THE PROFILE AND FIND FLOW PROPORTIES                     *
C*                                                                    *
C**********************************************************************
C*
      DO J = 1 , NUM_POINTS

        IF(J.NE.1)YY= YY + DY

        CALL HUNT(Y8,IC,YY,JLO)
        IF(JLO.LT.1)JLO = 1
        IF(JLO.GE.IC)JLO = IC - 1

        IF(J.EQ.1)THEN
         ZETA      = 0.0
        ELSE
         ZETA      = (YY-Y8(JLO)) / (Y8(JLO+1) - Y8(JLO))
        ENDIF

        IF(ZETA.LT.0.0)ZETA=0.0
c       write(6,*)'JLO,ZETA',JLO,ZETA

        Y9(J) = YY
        X9(J) = (1.-ZETA)*X8(JLO) + ZETA*X8(JLO+1) - X8(IC)
        IF(X9(J).LT.0.0)X9(J) = 0.0
        RHO9(J) = (1.-ZETA)*RHO8(JLO) + ZETA*RHO8(JLO+1)
        U9(J) = (1.-ZETA)*U8(JLO) + ZETA*U8(JLO+1)
        V9(J) = (1.-ZETA)*V8(JLO) + ZETA*V8(JLO+1)
        P9(J) = (1.-ZETA)*P8(JLO) + ZETA*P8(JLO+1)
        Q9(J) = SQRT(u9(J)**2 + V9(J)**2)
        A9(j) = ATAN(V9(J)/U9(J))

c       write(6,1020)J,J,X9(J),Y9(J),U9(J),V9(J),Q9(J),A9(J)*RAD,P9(J),RHO9(J)
      ENDDO
C*
C**********************************************************************
C*                                                                    *
C*    SMOOTH THE PROFILE AND FIND FLOW PROPORTIES                     *
C*                                                                    *
C**********************************************************************
C*
      I = 0
      WRITE(6,1000)

      DO J = 2*NI , 3*NI-1
       I = I + 1
```

```
       X(J) = X9(I)
       Y(J) = Y9(I)
       U(J) = U9(I)
       V(J) = V9(I)
       Q(J) = Q9(I)
       A(J) = A9(I)
       P(J) = P9(I)
       R(J) = RHO9(I)

C*
C*********************************************************************
C*                                                                   *
C*    FIND AREA - Correct for planar/axisymmetric flow               *
C*       DELTA = 0 for planar, 1 for axisymmetric                    *
C*                                                                   *
C*********************************************************************
C*
       IF(J.NE.2*NI)THEN
        AREA = PI * (Y(J)**2 - Y(J-1)**2) * DELTA + (1-DELTA)*(Y(J)-Y(J-1))
       ELSE
        AREA = 0.0
       ENDIF

       R_AVE  = (R(J) + R(J-1))/2.0
       U_AVE  = (U(J) + U(J-1))/2.0
       MDOT = MDOT + R_AVE*U_AVE*AREA
c      WRITE(6,*)'R_AVE,U_AVE,AREA',R_AVE,U_AVE,AREA,MDOT
       F    = F +((P(J)-PA)+R_AVE*U_AVE**2/GC)*AREA

       Q1 = Q(J)
       P1 = P(J)
       R1 = R(J)

       CALL THERMO (Q1,P1,R1,T,C,M)

       JJ = NI-I+1 ! axial index
C*
C*********************************************************************
C*                                                                   *
C*    OUTPUT THE FLOW VARIABLES                                      *
C*                                                                   *
C*********************************************************************
C*
       WRITE(6,2001)I,JJ,X(J),Y(J),U(J),V(J),M,Q(J),A(J)*RAD,P(J),R(J),T
       WRITE(13,2001)I,JJ,X(J),Y(J),U(J),V(J),M,Q(J),A(J)*RAD,P(J),R(J),T
C*
C*********************************************************************
C*                                                                   *
C*    PLACE THE FLOW VARIABLES INTO STORAGE                          *
C*                                                                   *
C*********************************************************************
C*
       XY(I,J,1)   = X(J)
       XY(I,J,2)   = Y(J)
       FLOW(I,J,1) = U(J)
       FLOW(I,J,2) = V(J)
       FLOW(I,J,3) = M
       FLOW(I,J,4) = Q(J)
       FLOW(I,J,5) = A(J)
       FLOW(I,J,6) = P(J)
       FLOW(I,J,7) = R(J)
       FLOW(I,J,8) = T

      ENDDO
C*
C*********************************************************************
C*                                                                   *
C*    FIND MASS FLOW AND FORCE                                       *
C*                                                                   *
C*********************************************************************
C*
      YT    = Y(3*NI-1)
      AT    = PI*YT*YT/GL*DELTA + (1-DELTA)*YT
      AS    = SQRT(2.0*G*GC*RG*TS/(G+1.0))
      MDOTS = AS*RR(AS)*AT
      FS    = PP(AS)*AT+MDOTS*AS/GC
      FOD   = FS-PA*AT
      CD   = MDOT/MDOTS
      ETAF = F/FOD
      ETAI = ETAF/CD
```

```
      WRITE(6,*)' '
      WRITE(6,2002)MDOT,MDOTS,CD,F,FOD,ETAF,ETAI
C
C***********************************************************************
C*                                                                     *
C*    MAKE GNU PLOT FOR VIEWING DATA                                   *
C*                                                                     *
C***********************************************************************
C
      IF(DEBUG)THEN

      WRITE(6,*)' '
      OPEN(UNIT=55,FILE='STARTING_CONDITIONS.dem',FORM='FORMATTED')
      WRITE(55,500)
 500  FORMAT('# GNUPLOT v3.6 beta multiplot script file',/,
     >'set terminal pdf enhanced color font "Helvetica" fsize 8 size 10.5 in, 8.0 in',/,
     >'set output "STARTING_CONDITIONS.pdf"',/,
     >'set key left top box',/,
     >'set border',/,
     >'set grid',/,
     >'set style line 1 lt rgb "black"   lw 1 pt 1 ps 1',/,
     >'set style line 2 lt rgb "red"     lw 1 pt 2 ps 1',/,
     >'set style line 3 lt rgb "green"   lw 1 pt 3 ps 1',/,
     >'set style line 4 lt rgb "blue"    lw 1 pt 4 ps 1',/,
     >'set style line 5 lt rgb "orange"  lw 1 pt 5 ps 1',/,
     >'set style line 6 lt rgb "yellow"  lw 1 pt 6 ps 1',/,
     >'show terminal ',/,
     >'#set title "STARTING NOZZLE FLOW CONDITIONS"',/,
     >'set ylabel "Radial Length"',/,
     >'set autoscale x',/,
     >'set autoscale y',/,
     >'set multiplot layout 2, 1 title "STARTING NOZZLE FLOW CONDITIONS"',/,
     >'set size 0.5,0.5',/,
     >'set origin 0,0',/,'set xlabel "Density (kg/m^3)"',/,
     >'plot [:] [:] "fort.7" using 11:4 with lines title "Post-Processed" ls 1, ',
     >'"Profile.dat" using  7:5 with lines title "Pre-Processed" ls 2',/,
     >'set size 0.5,0.5',/,
     >'set origin 0,0.49',/,'set xlabel "Pressure (Pa)"',/,
     >'plot [:] [:] "fort.7" using 10:4 with lines title "Post-Processed" ls 1, ',
     >'"Profile.dat" using 11:5 with lines title "Pre-Processed" ls 2',/,
     >'set size 0.5,0.5',/,
     >'set origin 0.5,0.49',/,'set xlabel "Temperature (K)"',/,
     >'plot [:] [:] "fort.7" using 12:4 with lines title "Post-Processed" ls 1, ',
     >'"Profile.dat" using 12:5 with lines title "Pre-Processed" ls 2',/,
     >'set size 0.5,0.5',/,
     >'set origin 0.5,0.0',/,'set xlabel "MACH NUMBER"',/,
     >'plot [:] [:] "fort.7" using 7:4 with lines title "Post-Processed" ls 1, ',
     >'"Profile.dat" using 13:5 with lines title "Pre-Processed" ls 2')

      CLOSE(55)

      WRITE(6,*)' '
      WRITE(6,*)'  Type: gnuplot STARTING_CONDITIONS.dem ; open STARTING_CONDITIONS.pdf'

      ENDIF

C*
C***********************************************************************
C*                                                                     *
C*    RETURN TO MAIN PROGRAM                                           *
C*                                                                     *
C***********************************************************************
C*
      RETURN
C*
C***********************************************************************
C*                                                                     *
C*    FORMAT STATEMENTS                                                *
C*                                                                     *
C***********************************************************************
C*

  13  FORMAT('ERROR: File Profile.dat does not exist.')
 200  FORMAT(1P,8E15.5)
 1010 FORMAT(/,'    I    J ','   X        ', '     Y      ',' U VELOCITY ', ' V VELOCITY ',
     >'VELOCITY MAG',' FLOW ANGLE ',' PRESSURE   ','  DENSITY   ')
 1020 FORMAT(2I5,1P10E12.4)
 1000 FORMAT('    POINT  ','   X BAR    ', '     Y      ',' U VELOCITY ', ' V VELOCITY ',
     >'MACH NUMBER ', 'VELOCITY MAG',' FLOW ANGLE ',' PRESSURE   ','  DENSITY   ',' TEMPERATURE')
 2001 FORMAT(2I5,1P10E12.4)
 2002 FORMAT(/,1P,
```

```
     >       9X,'Mass Flow Rate      = ',E12.3,/,
     >       9X,'Mass Flow Rate (1D) = ',E12.3,/,
     >       9X,'Nozzle Discharge Coefficient = ',E12.3,/,
     >       /,
     >       9X,'Thrust                     = ',E12.3,/,
     >       9X,'Thrust (1D)                = ',E12.3,/,
     >       9X,'Thrust ratio (efficiency) = ',E12.5,/,
     >       9X,'Thrust ratio / Nozzle Discharge = ',E12.5)
       END

      SUBROUTINE OUTPUT
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'

      WRITE(6,100)' MACHLINE + ANGLE:  = ',LP*RAD
      WRITE(6,100)' MACHLINE - ANGLE:  = ',LM*RAD
      WRITE(6,*)' '
      WRITE(6,100)' STREAMLINE ANGLE:  = ',L0*RAD
      WRITE(6,*)' '
      WRITE(6,100)' POINT 1 (X,Y):', X1,Y1
      WRITE(6,100)' PRESSURE        =', P1
      WRITE(6,100)' DENSITY         =', R1
      WRITE(6,100)' ANGLE, Degrees  =',A1*RAD
      WRITE(6,100)' VELOCITY        =',Q1
      WRITE(6,*)' '
      WRITE(6,100)' POINT 2 (X,Y):', X2,Y2
      WRITE(6,100)' PRESSURE        =', P2
      WRITE(6,100)' DENSITY         =', R2
      WRITE(6,100)' ANGLE, Degrees  =',A2*RAD
      WRITE(6,100)' VELOCITY        =',Q2
      WRITE(6,*)' '
      WRITE(6,100)' POINT 3 (X,Y):', X3,Y3
      WRITE(6,100)' PRESSURE        =', P3
      WRITE(6,100)' DENSITY         =', R3
      WRITE(6,100)' ANGLE, Degrees  =',A3*RAD
      WRITE(6,100)' VELOCITY        =',Q3
      WRITE(6,*)' '
      WRITE(6,100)' POINT 4 (X,Y):', X4,Y4
      WRITE(6,100)' PRESSURE        =', P4
      WRITE(6,100)' DENSITY         =', R4
      WRITE(6,100)' ANGLE, Degrees  =',A4*RAD
      WRITE(6,100)' VELOCITY        =',Q4

 100  FORMAT(A,1P2E15.5)

      RETURN
      END
      SUBROUTINE TURNING_CONTOUR
C
C*********************************************************************
C*                                                                   *
C*     TERMINOLOGY FOR SUPERSONIC FLOW METHOD OF CHARACTERISTICS      *
C*                                                                   *
C* CONTROL VARIABLE:                                                 *
C* ----------------                                                  *
C*                                                                   *
C* DELTA  = '0' FOR PLANER FLOW                                      *
C*        = '1' AXISYMMETRIC FLOW                                    *
C* ICOR   = NUMBER OF APPLICATIONS OF THE CORRECTOR DESIRED          *
C* E1     = CONVERGENCE TOLERANCE FOR LOCATION, M (IN)               *
C* E2     = CONVERGENCE TOLERANCE FOR VELOCITY, M/S (FT/SEC)         *
C* GC     = 1.0 M-KG/N-S^2 OR 32.174 FT-LBM/LBF-S^2                  *
C* GL     = 1.0 M^2/M^2 OR 144.0 IN^2/FT^2                           *
C* ST     = 0.0 EVEN SPACE STARTLINE, 1.0 QUADRATIC SPACED           *
C*                                                                   *
C* GAS THERMODYNAMIC PROPERTIES & STAGNATION PROPERTIES:             *
C* ----------------------------------------------------              *
C*                                                                   *
C* G      = RATIO OF SPECIFIC HEATS                                  *
C* RG     = GAS CONSTANT, J/KG-K (FT-LBF/LBM-R)                      *
C* TS     = STAGNATION TEMPERATURE, K (R)                            *
C* PS     = STAGNATION PRESSURE, N/M^2 (LBF/IN^2)                    *
C* PA     = AMBIENT PRESSURE, N/M^2 (LBF/IN^2)                       *
C*                                                                   *
C* FLOW FIELD PROPERTIES:                                            *
C* ---------------------                                             *
C*                                                                   *
C* X      = AXIAL COORDINATE, M (IN)                                 *
C* Y      = RADIAL COORDINATE, M (IN)                                *
C* U      = AXIAL VELOCITY, M/S (FT/S)                               *
C* V      = RADIAL VELOCITY, M/S (FT/S)                              *
```

```
C* Q       = VELOCITY MAGNITUDE, M/W (FT/S)                                *
C* A       = FLOW ANGLE, RAD                                               *
C* P       = STATIC PRESSURE, N/M^2 (LBF/IN^2)                             *
C* R       = STATIC DENSITY, KG/M^3 (LBM/FT^3)                             *
C* T       = STATIC TEMPERATURE, K (R)                                     *
C* C       = SPEED OF SOUND, M/S (FT/S)                                    *
C* M       = MACH NUMBER                                                   *
C* EMD     = DESIGN MACH NUMBER                                            *
C* 1,2,3,  = DENOTES PROPERTIES AT POINTS                                  *
C*                                                                         *
C* TERMINOLOGY EMPLOYED:                                                   *
C* --------------------                                                    *
C*                                                                         *
C* L       = TAN(THETA+-ALPHA)                                             *
C* Q       = (U^2-C^2), M^2/S^2 (FT^2/S^2)                                 *
C* R       = 2UV-L(U^2-C^2) M^2/S^2 (FT^2/S^2)                             *
C* S       = DELTA*C^2*V/Y, M^2/S^3 (FT^2/SEC^3-IN)                        *
C* T       = S*DEL(X)+Q*U+R*V, M^3/S^3 (FT^3/S^3)                          *
C* +/-     = DENOTES + OR - CHARACTERISTIC CURVE                           *
C*                                                                         *
C***************************************************************************
C
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C
C**************************************************************************
C**                                                                      **
C**   Function Statements                                                **
C**                                                                      **
C**************************************************************************
C
      TM(B) = B / TZERO
      ZMUIFD(B) = ZMZERO*TM(B)**1.5*((1.+CONSU)/(TM(B)+CONSU))
      RE(B) = R(1) * Q(1) * X(1) / ZMUIFD(B)
      BL(B) = RE(B)**(1./8.)
C
C***************************************************************************
C*                                                                         *
C*    OUTPUT THE HEADER                                                    *
C*                                                                         *
C***************************************************************************
C
      WRITE(6,*)' '
      ASTR = 'TURNING CONTOUR REGION'
      CALL BANNER(ASTR)
C
C***************************************************************************
C*                                                                         *
C*    CALCULATE THE TURNING CONTOUR REGION                                 *
C*                                                                         *
C***************************************************************************
C
c     IMAX = I+I_FINAL
      I = IMAX
      J = J2
C
C***************************************************************************
C*                                                                         *
C*    START MARCHING DOWNSTREAM                                            *
C*                                                                         *
C***************************************************************************
C
      DO 400 II = I+1 , I+I_FINAL
C
C***************************************************************************
C*                                                                         *
C*    OUTPUT THE HEADER                                                    *
C*                                                                         *
C***************************************************************************
C
       WRITE(11,*)' '
       WRITE(6,1010)
C
C***************************************************************************
C*                                                                         *
C*    SET THE CONSTANTS                                                    *
C*                                                                         *
C***************************************************************************
C
c      L  = L + 1
       YW = Y(1)
```

```
       PW = P(1)
       MFR  = 0.0
       MFRT = 0.0
       MFRT_OLD = 0.0
C
C*********************************************************************
C*                                                                   *
C*    BEGIN MARCHING UP FROM THE FINAL C+ CHARACTERISTIC LINE TO THE  *
C*    WALL.                                                          *
C*                                                                   *
C*********************************************************************
C
       DO 402 JJ= J , 1, -1

        IF(JJ.EQ.J)THEN                !On FINAL C+ CHARACTERISTIC LINE
         CALL MOVE(4,J)
         X(J)    = XY(II,J,1)
         Y(J)    = XY(II,J,2)
         X4      = X(J)
         Y4      = Y(J)
         R4      = R(J)
         A4      = A(J)
         U4      = Q1*COS(A4)
         V4      = Q1*SIN(A4)

         CALL THERMO(Q4,P4,R4,T4,C,M4)

C        AREA_X = PI * Y(JJ)**2
         AREA_X = PI * Y(JJ)**2 * DELTA + (1-DELTA)*Y(JJ)
         MFRT   = R4*U4*AREA_X

        ELSE                           !ON INTERIOR POINT TURNING SECTION

         CALL MOVE(1,JJ)               ! LOAD THE CURRENT POINT INTO P1
         CALL MOVE(2,JJ+1)             ! LOAD THE Final Characteristic Line CONDITIONS INTO P2
C                                      !  OR THE OLD P4 POINT
         U2 = Q2*COS(A2)
         V2 = Q2*SIN(A2)
         CALL THERMO(Q2,P2,R2,T2,C,M2)
c                                      !LOAD THE NEXT POINT INTO P3 BUT MAKE SURE IT EXIST
         IF(JJ.EQ.1)THEN               !IF JJ=1 THEN ITS A WALL POINT AND THEN P3=P1=P5
          CALL MOVE(3,JJ)
         ELSEIF(XY(II-1,JJ-1,1).EQ.0.0 .AND. XY(II-1,JJ-1,2).EQ.0.0)THEN    !Still a wall point
          CALL MOVE(3,JJ)
         ELSE
          CALL MOVE(3,JJ-1)              !IF JJ=1 THEN ITS A WALL POINT AND THEN P3=P1=P5
         ENDIF

         CALL TURNING

         CALL MOVE(5,JJ)               !SAVE THE RESULTS INTO POINT P4

C*
C*********************************************************************
C*                                                                   *
C*    FIND THE MASS FLUX                                             *
C*                                                                   *
C*********************************************************************
C*
       IF(JJ.EQ.J)THEN
        AREA_X = PI * Y4**2 * DELTA + (1-DELTA)*Y4
        AREA_Y = 0.0
        DEL_X  = 0.0
        DEL_Y  = 0.0
        R_AVE  = R4
        U_AVE  = U4
        V_AVE  = 0.0
       ELSE
        DEL_X  = ABS(X4-X(JJ+1))
        DEL_Y  = ABS(Y4**2-Y(JJ+1)**2)*DELTA + (1-DELTA)*ABS(Y4-Y(JJ+1))
        AREA_X = PI * DEL_Y * DELTA + (1-DELTA)*DEL_Y
        AREA_Y = 2.0 * PI * (Y4 + Y(JJ+1))/2.0*DEL_X * DELTA + (1-DELTA)*DEL_X
        R_AVE  = (R4 + R(JJ+1))/2.0
        U_AVE  = (U4 + U(JJ+1))/2.0
        V_AVE  = (V4 + V(JJ+1))/2.0
       ENDIF

c       WRITE(6,*)'R4,R2,R',R4,R2,R_AVE
c       WRITE(6,*)'U4,U2,U',U4,U2,U_AVE
c       WRITE(6,*)'V4,V2,V',V4,V2,V_AVE
c       WRITE(6,*)'AREA_X,AREA_Y',AREA_X,AREA_Y
```

```
        MFR     = R_AVE*(U_AVE*AREA_X+V_AVE*AREA_Y)
        MFRT    = MFR + MFRT
C*
C*********************************************************************
C*                                                                   *
C*    CHECK THE MASS FLUX                                            *
C*                                                                   *
C*********************************************************************
C*
        ENDIF

        IF(MFRT.GE.MDOT)THEN
c          ZETA = SQRT((MDOT - MFRT_OLD) / (MFRT - MFRT_OLD))
         ZETA = (SQRT(MDOT) - SQRT(MFRT_OLD)) / (SQRT(MFRT) - SQRT(MFRT_OLD))
         X4 = X4*ZETA + (1.-ZETA)*X(JJ+1)
         Y4 = Y4*ZETA + (1.-ZETA)*Y(JJ+1)
c          WRITE(6,*)'ZETA = ',ZETA,X4,Y4
c          CALL FIND_MASS_LINE(JJ+1,MFRT_OLD)
         M4 = M4*ZETA + (1.-ZETA)*M2
         Q4 = Q4*ZETA + (1.-ZETA)*Q(JJ+1)
         A4 = A4*ZETA + (1.-ZETA)*A(JJ+1)
         U4 = Q4*COS(A4)
         V4 = Q4*SIN(A4)
         P4 = P4*ZETA + (1.-ZETA)*P(JJ+1)
         R4 = R4*ZETA + (1.-ZETA)*R(JJ+1)
         T4 = T4*ZETA + (1.-ZETA)*T2
         CALL MOVE(5,JJ)
c        WRITE(6,*)'MFRT,MDOT,MFRT_OLD,ZETA',MFRT,MDOT,MFRT_OLD,ZETA
        ENDIF

        MFRT_OLD = MFRT
C*
C*********************************************************************
C*                                                                   *
C*    OUTPUT THE FLOW VARIABLES                                      *
C*                                                                   *
C*********************************************************************
C*
        AP = A4*RAD
        PP = P4/GL
c       LJ = L + J

        IF (KW.EQ.1)THEN
         WRITE(6,1020)II,JJ,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
c         IF(MFRT.GE.MDOT)write(6,*)' MASS FLUX = ',MFRT
         WRITE(11,1020)II,JJ,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
         IF(MFRT.GE.MDOT)WRITE(3,1020)II,JJ,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4

IF(MFRT.GE.MDOT)WRITE(32,1020)II,JJ,X4,Y4,TM(T4),ZMUIFD(T4),RE(T4),BL(T4),Y4+BL_SCALE*0.1404*((1.
+X4)**0.125-1.0)
c         IF(J.EQ.J2.AND.Y4.EQ.0.0.AND.X4.LE.XE)WRITE(4,1020)II,JJ,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
        ENDIF
C*
C*********************************************************************
C*                                                                   *
C*    PLACE THE FLOW VARIABLES INTO STORAGE                          *
C*                                                                   *
C*********************************************************************
C*
         XY(II,JJ,1)    = X4
         XY(II,JJ,2)    = Y4
         IF(Y4.LE.XY(II,JJ+1,2))THEN
           WRITE(6,*)'Error in Y!'
         ENDIF
         FLOW(II,JJ,1) = U4
         FLOW(II,JJ,2) = V4
         FLOW(II,JJ,3) = M4
         FLOW(II,JJ,4) = Q4
         FLOW(II,JJ,5) = A4
         FLOW(II,JJ,6) = P4
         FLOW(II,JJ,7) = R4
         FLOW(II,JJ,8) = T4

        IF(MFRT.GE.MDOT)GOTO 409
C*
C*********************************************************************
C*                                                                   *
C*    END OF THE J INDEX LOOP - MARCHING TO THE WALL LOOP            *
C*                                                                   *
C*********************************************************************
```

```
C*
 402   ENDDO

 409   CONTINUE
C*
C*********************************************************************
C*                                                                   *
C*    LOAD THE FINAL C+ CHARACTERISTIC LINE DATA INTO POINT P1        *
C*                                                                   *
C*********************************************************************
C*
       U1 = FLOW(I,J,1)
       V1 = FLOW(I,J,2)
       M1 = FLOW(I,J,3)
       Q1 = FLOW(I,J,4)
       A1 = FLOW(I,J,5)
       A4 = A1
       P1 = FLOW(I,J,6)
       R1 = FLOW(I,J,7)
       T1 = FLOW(I,J,8)
C*
C*********************************************************************
C*                                                                   *
C*    END OF THE I INDEX LOOP - DOWNSTREAM MARCHING LOOP             *
C*                                                                   *
C*********************************************************************
C*
 400  ENDDO
C
C*********************************************************************
C*                                                                   *
C*    RETURN                                                         *
C*                                                                   *
C*********************************************************************
C
      RETURN
C
C*********************************************************************
C*                                                                   *
C*    FORMAT STATEMENTS                                              *
C*                                                                   *
C*********************************************************************
C
 1010 FORMAT(/,'    I    J ','   X        ', '     Y      ',' U VELOCITY ', ' V VELOCITY ',
     >'MACH NUMBER ', 'VELOCITY MAG',' FLOW ANGLE ',' PRESSURE   ','  DENSITY   ',' TEMPERATURE')
 1020 FORMAT(2I5,1P10E12.4)
C
C*********************************************************************
C*                                                                   *
C*    END OF LINE                                                    *
C*                                                                   *
C*********************************************************************
C
      END
      SUBROUTINE DIRECT_WALL_POINT
C...
C...  Points 3 and 2 are known
C...  Point 4 is located on the wall
C...  Point 3 is located on the wall
C...  The C- characteristic goes through points 3 and 2
C...  The C+ characteristic goes through points 2 and 4
C...  The streamline        goes through points 3 and 4
C...
C...
C...               ##
C...   WALL      ##
C...          ## *4
C...       ##   *
C...    ##     *
C... ##       * C+
C...  3*     *
C...    *   *
C...  C- * *
C...     2*


C...
C... CALCULATE THE SOLUTION AT A DIRECT WALL POINT
C...  POINTS 2 AND 3 ARE KNOWN, SOLVE FOR POINT 4 ALONG WALL USING CHARA+
C...
      INCLUDE 'MOC.PAR'
```

```
      INCLUDE 'MOC.CMN'
C...
C... CALCULATE THE COEFFICIENTS FOR THE PREDICTOR
C...
      ITER = 0

      Q4 = Q3
      P4 = P3
      R4 = R3

      CALL THERMO (Q2,P2,R2,T,C,M)

      LP = TAN(A2+ASIN(1./M))
      QP = GC*SQRT(M**2-1.)/(R2*Q2**2)
      SP = DELTA*SIN(A2)/(Y2*M*COS(A2+ASIN(1./M)))
C...
C... WE NEED TO FIND POINT X4
c...
 10   CALL BOUNDARY(2)
C...
C... SOLUTION
C...
      TP = -SP*(X4-X2)+QP*P2+A2

      P0 = 0.5*(P3+P4)
      R8 = 0.5*(R3+R4)
      Q0 = 0.5*(Q3+Q4)
      R0 = R8*Q0/GC

      CALL THERMO (Q0,P0,R8,T,C,M)

      A0 = C**2/GC
      T01 = R0*Q3+P3
      T02 = P3-A0*R3

      P4 = (TP-A4)/QP
      Q4 = (T01-P4)/R0
      R4 = (P4-T02)/A0
C...
C... TEST FOR CONVERGENCE
C...

c     WRITE(6,*)' '
c     WRITE(6,100)' R0                 = ',R0
c     WRITE(6,100)' A0                 = ',A0
c     WRITE(6,100)' T01                = ',T01
c     WRITE(6,100)' T02                = ',T02
c     WRITE(6,100)' Q+                 = ',QP
c     WRITE(6,100)' S+                 = ',SP
c     WRITE(6,100)' T+                 = ',TP
 100  FORMAT(A,1P2E15.5)

      IF(ITER.EQ.ICOR) RETURN
      IF(ITER.EQ.0)    GOTO 20

      IF((ABS(X4-XD).GT.E1)     .OR.  (ABS(Y4-YD).GT.E1))    GOTO 20
      IF((ABS(P4-PD).GT.E2*PD)  .OR.  (ABS(R4-RD).GT.E3*RD)) GOTO 20
      IF((ABS(Q4-QD).GT.E4*QD)  .AND. (ABS(A4-AD).GT.E5*AD)) RETURN
C...
C... CALCULATE THE COEFFICIENTS FOR THE CORRECTOR
C...
 20   ITER = ITER + 1

      XD = X4
      YD = Y4
      PD = P4
      RD = R4
      QD = Q4
      AD = A4

      P0 = 0.5*(P2+P4)
      R8 = 0.5*(R2+R4)
      Q0 = 0.5*(Q2+Q4)
      A0 = 0.5*(A2+A4)
      Y0 = 0.5*(Y2+Y4)

      CALL THERMO (Q0,P0,R8,T,C,M)

      LP = TAN(A0+ASIN(1.0/M))
      SP = DELTA*SIN(A0)/(Y0*M*COS(A0+ASIN(1.0/M)))
      QP = GC*SQRT(M**2-1.0)/(R8*Q0**2)
```

```
      GOTO 10
      END
      SUBROUTINE LEAST_SQR_FIT(X,Y,MM,K,R)
c*******************************************************
c*  LEAST SQUARE APPROXIMATION OF A DISCREET FUNCTION   *
c*  USING ORTHOGONAL POLYNOMIALS                        *
c* -------------------------------------------------- *
c*******************************************************
      REAL X(MM),Y(MM),SIGMA(MM),S(0:K),ECART(0:K),ALPHA(K),
     >   BETA(0:K-1),P,R(MM),W,W0,DW,PI,WORK(4*MM)

      SIGMA = 1.0

      CALL  MCAPPR(K,MM,X,Y,SIGMA,S,ALPHA,BETA,ECART,WORK)

      PRINT *,' '
      WRITE(*,'(1X,A)') '  I     COEFFICIENTS    STD DEVIATION'
      WRITE(*,'(I4,2E17.8)') (I,S(I),ECART(I),I=0,K)
C         PRINT *,' '
c     WRITE(*,'(1X,A)') '  I     VARIABLE         EXACT R          APPROX. R'

      DO I=1,MM
      W=X(I)
      R(I)=P(K,S,ALPHA,BETA,W)
c     WRITE(*,'(I4,3E17.8)') I,Y(I),X(I),R(I)
      ENDDO

      RETURN
      END

      SUBROUTINE MCAPPR(K,M,X,Y,SIGMA,S,ALPHA,BETA,ECART,WORK)
c======================================================================
c     LEAST SQUARES APPROXIMATION OF A FUNCTION F(X) DEFINED BY M POINTS
c     X(I), Y(I) BY USING ORTHOGONAL POLYNOMIALS
c======================================================================
c     INPUTS:
c     K   : DEGREE OF POLYNOMIALS
c     M   : NUMBER OF POINTS
c     X,Y : TABLES OF DIMENSION M TO STORE M ABSCISSAS AND
c           M ORDINATES OF GIVEN POINTS
```

```
c      SIGMA : TABLE OF DIMENSION M TO STORE THE STANDARD DEVIATIONS
c              OF VARIABLE Y
c      OUTPUTS:
c      S     : TABLE OF DIMENSION(0:K)
c      ALPHA: TABLE OF DIMENSION (K)
c      BETA : TABLE OF DIMENSION (0:K-1)
c      WORKING SPACE:
c      WORK : TABLE OF DIMENSION (4*M)
c      NOTE:
c      COEFFICIENTS S,ALPHA,BETA ARE USED TO EVALUATE VALUE
c      AT POINT Z OF THE BEST POLYNOMIAL OF DEGREE K
c      BY USING FUNCTION P(K,S,ALPHA,BETA,Z) DESCRIBED BELOW.
c=======================================================================
       REAL X(M),Y(M),SIGMA(M),S(0:K),ECART(0:K),ALPHA(K),
     >    BETA(0:K-1), WORK(4*M)
       M1=M+1
       M2=M+M1
       M3=M+M2
       CALL MCARRE(K,M,X,Y,SIGMA,S,ALPHA,BETA,ECART,WORK(1),WORK(M1),
     >            WORK(M2),WORK(M3))
       END

       !LEAST SQUARES SUBROUTINE
       SUBROUTINE MCARRE(K,M,X,Y,SIGMA,S,ALPHA,BETA,ECART,P1,P2,P3,P4)
c      IMPLICIT REAL(A-H,O-Z)
       DIMENSION X(M),Y(M),SIGMA(M),S(0:K),ECART(0:K),ALPHA(K),
     >BETA(0:K-1),P1(M),P2(M),P3(M),P4(M)
       DO I=1,M
       P1(I)=0.D0
       P2(I)=1.D0
       ENDDO
       W=M
       BETA(0)=0.D0
       DO I=0,K-1
       OMEGA=PRD(Y,P2,SIGMA,M)
       S(I)=OMEGA/W
       T=PRD(P2,P2,SIGMA,M)/W**2
       ECART(I)=SQRT(T)
       DO L=1,M
       P4(L)=X(L)*P2(L)
```

```
      ENDDO
      ALPHA(I+1)=PRD(P4,P2,SIGMA,M)/W
      DO L=1,M
      P3(L)=(X(L)-ALPHA(I+1))*P2(L)-BETA(I)*P1(L)
      ENDDO
      WPR=PRD(P3,P3,SIGMA,M)
      IF(I+1.LE.K-1) BETA(I+1)=WPR/W
      W=WPR
      DO L=1,M
      P1(L)=P2(L)
      P2(L)=P3(L)
      ENDDO
      ENDDO
      OMEGA=PRD(Y,P2,SIGMA,M)
      S(K)=OMEGA/W
      T=PRD(P2,P2,SIGMA,M)/W**2
      ECART(K)=SQRT(T)
      RETURN
      END
      FUNCTION PRD(X,Y,Z,M)
      REAL*8 X(M),Y(M),Z(M),PRD,SUM
      SUM=0.D0
      DO I=1,M
      SUM=SUM+X(I)*Y(I)/Z(I)**2
      ENDDO
      PRD=SUM
      RETURN
      END

      FUNCTION P(K,S,ALPHA,BETA,X)
C======================================================================
c     THIS FUNCTIONE ALLOWS EVALUATING VALUE AT POINT X OF A FUNCTION
c     F(X), APPROXIMATED BY A SYSTEM OF ORTHOGONAL POLYNOMIALS Pj(X)
c     THE COEFFICIENTS OF WHICH, ALPHA,BETA HAVE BEEN DETERMINED BY
c     LEAST SQUARES.
C======================================================================
      REAL S(0:*),ALPHA(*),BETA(0:*),P,B,X,BPR,BSD
      B=S(K)
      BPR=S(K-1)+(X-ALPHA(K))*S(K)
      DO I=K-2,0,-1
```

```
      BSD=S(I)+(X-ALPHA(I+1))*BPR-BETA(I+1)*B

      B=BPR

      BPR=BSD

      ENDDO

      P=BPR

      RETURN

      END

      SUBROUTINE FINAL_CHAR_LINE
C
C**********************************************************************
C*                                                                    *
C*    FINAL C+ CHARACTERISTIC LINE CALCULATION                        *
C*                                                                    *
C**********************************************************************
C
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C*
C**********************************************************************
C*                                                                    *
C*    OUTPUT STATEMENTS                                               *
C*                                                                    *
C**********************************************************************
C*
      WRITE(6,*)' '
      ASTR = 'REACHED DESIGNED MACH NUMBER ON CENTERLINE'
      CALL BANNER(ASTR)

      WRITE(6,*)' '
      WRITE(6,*)'   FINAL C+ CHARACTERISTIC LINE CALCULATION'
      WRITE(6,*)' '
C*
C**********************************************************************
C*                                                                    *
C*    FIND THE NEW STOPPING LOCATION BASED ON THE MASS FLOW RATE      *
C*                                                                    *
C**********************************************************************
C*
      MDOT = MFRT        !This seems to be a smart thing to do
      WRITE(6,*)' '
      WRITE(6,*)' RESETTING MASS FLUX = ',MDOT
      IF(DELTA.EQ.1)THEN
       YE = SQRT(MDOT/R4/U4/PI)
      ELSE
       YE = MDOT/R4/U4
      ENDIF
      XE = X4 + YE/TAN(ASIN(1./M4))

      WRITE(6,*)' '
      WRITE(6,*)'   NOZZLE EXIT POINT ',XE,YE
      WRITE(6,1010)

c     WRITE(8,*)' '
C*
C**********************************************************************
C*                                                                    *
C*    DEFINE THE POINTS ALONG THE FINAL C+ CHARACTERISTIC LINE        *
C*                                                                    *
C**********************************************************************
C*
      I = IMAX
      J = J2

      IF(I+I_FINAL.GT.NII)THEN
       WRITE(6,*)' '
       WRITE(6,*)' STOPPING CODE: RESET NII TO ',I+I_FINAL
       STOP
      ENDIF

      DO II = I , I+I_FINAL

       ZETA  = (II*1. - I*1. ) / ( I*1.+I_FINAL*1. - I*1. )

C...Even Spacing for y
```

```
       Y5     = YE*ZETA
       X5     = XE*ZETA + (1-ZETA)*X4

C...Quaddratic spacing for y
c      Y5     = YE*(ZETA)**2
c      X5     = XE*ZETA**2 + X4*(1.-ZETA**2)

C*
C***********************************************************************
C*                                                                     *
C*    OUTPUT THE FLOW VARIABLES                                        *
C*                                                                     *
C***********************************************************************
C*
       A4 = 0.0
       AP = A4*RAD
       PP = P4/GL
       WRITE(6,1020)II,J,X5,Y5,U4,V4,M4,Q4,AP,PP,R4,T4
       WRITE(10,1020)II,J,X5,Y5,U4,V4,M4,Q4,AP,PP,R4,T4
C*
C***********************************************************************
C*                                                                     *
C*    PLACE THE FLOW VARIABLES INTO STORAGE                            *
C*                                                                     *
C***********************************************************************
C*
       XY(II,J,1)   = X5
       XY(II,J,2)   = Y5
       FLOW(II,J,1) = U4
       FLOW(II,J,2) = V4
       FLOW(II,J,3) = M4
       FLOW(II,J,4) = Q4
       FLOW(II,J,5) = A4
       FLOW(II,J,6) = P4
       FLOW(II,J,7) = R4
       FLOW(II,J,8) = T4
C*
C***********************************************************************
C*                                                                     *
C*    END OF THE I INDEX LOOP                                          *
C*                                                                     *
C***********************************************************************
C*
      ENDDO
C
C***********************************************************************
C*                                                                     *
C*    RETURN TO THE MAIN CALLING PROGRAM                               *
C*                                                                     *
C***********************************************************************
C
      RETURN
C
C***********************************************************************
C*                                                                     *
C*    FORMAT STATEMENTS                                                *
C*                                                                     *
C***********************************************************************
C
 1010 FORMAT(/,'    I    J ','    X         ', '      Y       ',' U VELOCITY ', ' V VELOCITY ',
     >'MACH NUMBER ', 'VELOCITY MAG',' FLOW ANGLE ',' PRESSURE   ','  DENSITY    ',' TEMPERATURE')
 1020 FORMAT(2I5,1P10E12.4)
C
C***********************************************************************
C*                                                                     *
C*    END OF LINE                                                      *
C*                                                                     *
C***********************************************************************
C
      END
      SUBROUTINE LINE_INTERSECTION(AX,AY,BX,BY,CX,CY,DX,DY,H,G,IX,IY)
C...
C... THIS ROUTINE PROJECTS RAY AB ONTO RAY CD (H FACTOR) AND
C... THEN PROJECTS RAY CD ONTO AB (G FACTOR).  THE TWO RAYS
C... INTERSECT ONLY IF BOTH H & G ARE BEWTEEN 0 AND 1.
C...
C...if H is 0 then you are at point C
C...if H is 1 then you are at point D
C...
C...if G is 0 then you are at point A
C...if G is 1 then you are at point B
```

```
      REAL H,IX,IY

      EX = BX-AX
      EY = BY-AY

      FX = DX-CX
      FY = DY-CY

      GX = AX-CX
      GY = AY-CY

      PX = -EY
      PY = EX

      F_DOT_P = FX*PX + FY*PY
      G_DOT_P = GX*PX + GY*PY

      IF(F_DOT_P.NE.0.0)THEN
      H = G_DOT_P / F_DOT_P
      ELSE
      H = 1.0E+10
      ENDIF

c     WRITE(6,*)' H FACTOR: ',H
      IX = CX + FX*H
      IY = CY + FY*H
c     WRITE(6,*)' INTERSECT ON RAY CD (X,Y):',IX,IY

      EX = DX-CX
      EY = DY-CY

      FX = BX-AX
      FY = BY-AY

      GX = CX-AX
      GY = CY-AY

      PX = -EY
      PY = EX

      F_DOT_P = FX*PX + FY*PY
      G_DOT_P = GX*PX + GY*PY

      IF(F_DOT_P.NE.0.0)THEN
      G = G_DOT_P / F_DOT_P
      ELSE
      G = 1.0E+10
      ENDIF

c     WRITE(6,*)' G FACTOR: ',G
      IX = AX + FX*G
      IY = AY + FY*G
c     WRITE(6,*)' INTERSECT ON RAY AB (X,Y):',IX,IY
c     WRITE(6,*)' '
      IF(H.GE.0.0 .AND. H.LE.1.0 .AND.
     >   G.GE.0.0 .AND. G.LE.1.0 )THEN
c     WRITE(6,*)' RAYS INTERSECT'
      ELSE
c     WRITE(6,*)' RAYS DO NOT INTERSECT'
      ENDIF

      RETURN
      END
      SUBROUTINE READ_NAMELIST
C
C*********************************************************************
C*                                                                   *
C*    READ IN THE NAMELIST INPUT                                     *
C*                                                                   *
C*********************************************************************
C
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C...
      NAMELIST /INPUT/ DELTA,IUNITS,ICOR,E1,E2,E3,E4,E5,IORDER,
     >                 GC,GL,G,RG,CP,NI,NT,KWRITE,
     >                 PS,TS,PA,YT,RTU,RTD,AA,AE,XE,ST,
     >                 EMD,NOZ,SMOOTH,IVL,I_FINAL,BL_SCALE
C
C*********************************************************************
C*                                                                   *
```

```
C*    SET DEFAULT CONDITIONS                                            *
C*                                                                      *
C***********************************************************************
C
      IORDER  = 15
      BL_SCALE = 1.0
      TZERO   = 273.0
      CONSU   = 110.4/TZERO
      ZMZERO = .00001716
      I_FINAL = 50
      SMOOTH = .FALSE.
      EMD    = 3.0
      NOZ    = 0
      ERROR = 1.0E-10
      JMAX   = 0
      IVL    = 0
      XY     = 0.0
      FLOW   = 0.0
      X1     = 0.0
      X2     = 0.0
      X3     = 0.0
      X4     = 0.0
      Y1     = 0.0
      Y2     = 0.0
      Y3     = 0.0
      Y4     = 0.0
      Q1     = 0.0
      Q2     = 0.0
      Q3     = 0.0
      Q4     = 0.0
      A1     = 0.0
      A2     = 0.0
      P1     = 0.0
      P2     = 0.0
      P3     = 0.0
      P4     = 0.0
      R1     = 0.0
      R2     = 0.0
      R3     = 0.0
      R4     = 0.0
      PI     = 4.0*ATAN(1.0)
      RAD    = 180./PI
      DESIGNED_M = .FALSE.
      EXIT_END   = .FALSE.
C
C***********************************************************************
C*                                                                      *
C*    SET DEFAULT PROPERTIES                                            *
C*                                                                      *
C***********************************************************************
C
      E1     = 1.0E-06  !m
      E2     = 1.0E-06  !m/s
      E3     = 1.0E-06  !m/s
      E4     = 1.0E-06  !m/s
      E5     = 1.0E-06  !m/s

      GC     = 1.0      !M-KG/N-S^2
      GL     = 1.0      !M^2/M^2

      G      = 1.2
      RG     = 320.000  !J/KG-K
      ST     = 0.0      !even spaced start
C
C***********************************************************************
C*                                                                      *
C*    SET DEFAULT CONTROL VALUES                                        *
C*                                                                      *
C***********************************************************************
C
      DELTA = 1.0
      NI     = 21
      NT     = 15
      ICOR   = 30
      KWRITE= 1
      IUNITS= 2
C
C***********************************************************************
C*                                                                      *
C*    SET DEFAULT GAS DATA                                              *
C*                                                                      *
```

```
C*********************************************************************
C
      PS    = 70.0E+05
      TS    = 3000.0
      PA    = 0.0
      G     = 1.4
      RG    = 287.040  !J/KG-K
      CP    = G * RG / (G - 1.)
C
C*********************************************************************
C*                                                                   *
C*    SET DEFAULT NOZZLE WALL CONDITIONS                             *
C*                                                                   *
C*********************************************************************
C
      YT  = 1.000 !nozzle radius
      RTU = 2.000 !radius of curve upstream
      RTD = 0.500 !radius of curve downstream
      AA  = 10.0 !attachment angle
      AE  = 10.0 !exit angle
      XE  = 10.0 !exit location
C
C*********************************************************************
C*                                                                   *
C*    READ IN THE NAME LIST INPUT DATA                               *
C*                                                                   *
C*********************************************************************
C
      WRITE(6,*)' '
      ASTR = ' INPUT CONDITIONS'
      CALL BANNER(ASTR)

      REWIND(2)
      IFIND = 0
      DO WHILE(IFIND.EQ.0)
       READ(2,'(A)',END=34)NMLNAME
       IF (NMLNAME(2:7).EQ.'&INPUT') IFIND=1
      ENDDO
      BACKSPACE(2)
      READ(2,INPUT)
!     WRITE(6,INPUT)
      CLOSE(2)

      CP    = G * RG / (G - 1.)
      PS    = PS * GL
      PA    = PA * GL
      IF(NI.GT.99)NI=99
C
C*********************************************************************
C*                                                                   *
C*    OUTPUT THE NAMELIST                                            *
C*                                                                   *
C*********************************************************************
C
      WRITE(6,*)' '
      WRITE(6,100)'!... Error Conditions'
      WRITE(6,101)'ICOR     =',ICOR,'!Predictor Corrector Term (set to 10)'
      WRITE(6,102)'E1       =',E1,'!X Space Error'
      WRITE(6,102)'E2       =',E2,'!Pressure Error'
      WRITE(6,102)'E3       =',E3,'!Density Error'
      WRITE(6,102)'E4       =',E4,'!Velocity Error'
      WRITE(6,102)'E5       =',E5,'!Flow Angle Error'

      WRITE(6,*)' '
      WRITE(6,100)'!... Control Conditions'
      WRITE(6,102)'DELTA    =',DELTA,'!1 axi, 0 planer (Mass flux not working correctly)'
      WRITE(6,102)'ST       =',ST,'!0.0 Even space start line, 1.0 quadaratic spaced'
      WRITE(6,101)'NI       =',NI,'!NUMBER OF RADIAL POINTS ON INFLOW PLANE (Max 99)'
      WRITE(6,101)'NT       =',NT,'!NUMBER OF CIRULAR ARC POINTS (Should = AA), used only if
IVL=1'
      WRITE(6,103)'SMOOTH   =',SMOOTH,'!EVENLY SPACE OUT THE DATA AFTER EACH SECTION'

      WRITE(6,*)' '
      WRITE(6,100)'!... Gas Proporties'
      WRITE(6,102)'GC       =',GC,'!1.0 M-KG/N-S^2 Or 32.174 FT-LBM/LBF-S^2'
      WRITE(6,102)'GL       =',GL,'!1.0 M^2/M^2 Or 144.0 IN^2/FT^2'
      WRITE(6,102)'PS       =',PS,'!Pa Gas Chamber Pressure'
      WRITE(6,102)'PA       =',PA,'!Pa Ambient Pressure'
      WRITE(6,102)'TS       =',TS,'!K Gas Chamber Temperature'
      WRITE(6,102)'G        =',G,'!Gamma, Ratio of Specific Heats'
      WRITE(6,102)'RG       =',RG,'!J/kg-K Gas constant'
```

```
      WRITE(6,102)'CP      =',CP,'!J/kg-K Specific Heat At Constant Pressure'

      WRITE(6,*)' '
      WRITE(6,100)'!... Start Line Condition'
      WRITE(6,101)'IVL     =',IVL,'!0 Transonic Start, 1 User Defined Input From "Profile.dat"'

      WRITE(6,*)' '
      WRITE(6,100)'!... Nozzle Wall Proporties'
      WRITE(6,102)'YT      =',YT,'!Nozzle Radius'
      WRITE(6,102)'RTU     =',RTU,'!Radius Of Curveture Upstream Of Throat'
      WRITE(6,102)'RTD     =',RTD,'!Radius Of Curveture Downstream Of Throat'
      WRITE(6,102)'AA      =',AA,'!Attachment Angle - Angle At Which Flow Transition To MOC Wall'
      WRITE(6,102)'XE      =',XE,'!Exit Location'
      WRITE(6,102)'AE      =',XE,'!Exit Angle'


      WRITE(6,*)' '
      WRITE(6,100)'!... Nozzle Solution'
      WRITE(6,102)'EMD     =',EMD,'!Design Exit Mach Number'
      WRITE(6,101)'NOZ     =',NOZ,'!1 Nozzle Shape Inputed, 0 Nozzle Shape From MOC'
      WRITE(6,101)'I_FINAL =',I_FINAL,'!1 # Of Points On Final Characteristic Line'
      WRITE(6,102)'BL_SCALE=',BL_SCALE,'!Boundary Layer Scaler'

 100  FORMAT(A)
 101  FORMAT(" ",A,I15,",     ",A)
 102  FORMAT(" ",A,1PE15.5,",     ",A)
 103  FORMAT(" ",A,L15,",     ",A)

      RETURN
C
C***********************************************************************
C*                                                                     *
C*    ERROR SECTION                                                    *
C*                                                                     *
C***********************************************************************
C
  34  WRITE(6,*)'Can not find namelist &INPUT in file "NOZZLE.inp"'
      STOP


      END
      SUBROUTINE SECOND_ORDER_WALL_FLOWFIELD
C
C***********************************************************************
C*                                                                     *
C* CALCULATE THE FLOW FIELD FROM THE SECOND ORDER QUADRATIC WALL       *
C*                                                                     *
C***********************************************************************
C
C
C***********************************************************************
C*                                                                     *
C*     TERMINOLOGY FOR SUPERSONIC FLOW METHOD OF CHARACTERISTICS       *
C*                                                                     *
C* CONTROL VARIABLE:                                                   *
C* -----------------                                                   *
C*                                                                     *
C* DELTA  = '0' FOR PLANER FLOW                                        *
C*        = '1' AXISYMMETRIC FLOW                                      *
C* ICOR   = NUMBER OF APPLICATIONS OF THE CORRECTOR DESIRED            *
C* E1     = CONVERGENCE TOLERANCE FOR LOCATION, M (IN)                 *
C* E2     = CONVERGENCE TOLERANCE FOR VELOCITY, M/S (FT/SEC)           *
C* GC     = 1.0 M-KG/N-S^2 OR 32.174 FT-LBM/LBF-S^2                    *
C* GL     = 1.0 M^2/M^2 OR 144.0 IN^2/FT^2                             *
C* ST     = 0.0 EVEN SPACE STARTLINE, 1.0 QUADRATIC SPACED             *
C*                                                                     *
C* GAS THERMODYNAMIC PROPERTIES & STAGNATION PROPERTIES:               *
C* -----------------------------------------------------               *
C*                                                                     *
C* G      = RATIO OF SPECIFIC HEATS                                    *
C* RG     = GAS CONSTANT, J/KG-K (FT-LBF/LBM-R)                        *
C* TS     = STAGNATION TEMPERATURE, K (R)                              *
C* PS     = STAGNATION PRESSURE, N/M^2 (LBF/IN^2)                      *
C* PA     = AMBIENT PRESSURE, N/M^2 (LBF/IN^2)                         *
C*                                                                     *
C* FLOW FIELD PROPERTIES:                                              *
C* ---------------------                                               *
C*                                                                     *
C* X      = AXIAL COORDINATE, M (IN)                                   *
C* Y      = RADIAL COORDINATE, M (IN)                                  *
C* U      = AXIAL VELOCITY, M/S (FT/S)                                 *
```

```
C* V       = RADIAL VELOCITY, M/S (FT/S)                                  *
C* Q       = VELOCITY MAGNITUDE, M/W (FT/S)                               *
C* A       = FLOW ANGLE, RAD                                              *
C* P       = STATIC PRESSURE, N/M^2 (LBF/IN^2)                            *
C* R       = STATIC DENSITY, KG/M^3 (LBM/FT^3)                            *
C* T       = STATIC TEMPERATURE, K (R)                                    *
C* C       = SPEED OF SOUND, M/S (FT/S)                                   *
C* M       = MACH NUMBER                                                  *
C* EMD     = DESIGN MACH NUMBER                                           *
C* 1,2,3,  = DENOTES PROPERTIES AT POINTS                                 *
C*                                                                        *
C* TERMINOLOGY EMPLOYED:                                                  *
C* --------------------                                                   *
C*                                                                        *
C* L       = TAN(THETA+-ALPHA)                                            *
C* Q       = (U^2-C^2), M^2/S^2 (FT^2/S^2)                                *
C* R       = 2UV-L(U^2-C^2) M^2/S^2 (FT^2/S^2)                            *
C* S       = DELTA*C^2*V/Y, M^2/S^3 (FT^2/SEC^3-IN)                       *
C* T       = S*DEL(X)+Q*U+R*V, M^3/S^3 (FT^3/S^3)                         *
C* +/-     = DENOTES + OR - CHARACTERISTIC CURVE                          *
C*                                                                        *
C*************************************************************************
C
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C
C************************************************************************
C**                                                                    **
C**   Function Statements                                              **
C**                                                                    **
C************************************************************************
C
      TM(B) = B / TZERO
      ZMUIFD(B) = ZMZERO*TM(B)**1.5*((1.+CONSU)/(TM(B)+CONSU))
      RE(B) = R(1) * Q(1) * X(1) / ZMUIFD(B)
      BL(B) = RE(B)**(1./8.)
C*
C*************************************************************************
C*                                                                        *
C*    OUTPUT STATEMENTS                                                   *
C*                                                                        *
C*************************************************************************
C*
      WRITE(6,*)' '
      ASTR = 'CALCULATE THE FLOW FIELD FROM THE SECOND ORDER QUADRATIC WALL'
      CALL BANNER(ASTR)
C*
C*************************************************************************
C*                                                                        *
C*    SET THE CONSTANTS                                                   *
C*                                                                        *
C*************************************************************************
C*
      III = NI + NT    !ending i index from circular arc throat
      II  = III + 1    !STARTING I INDEX

      III = II+10000   !ending i index, set to just some large number
      L   = 0

c     J2  = J3 - IEND
c     J2  = J2 + 1 - IEND
C*
C*************************************************************************
C*                                                                        *
C*    CALL BOUNDARY AND FIND THE WALL ANGLE BETWEEN P2 AND P4             *
C*                                                                        *
C*************************************************************************
C*
      CALL BOUNDARY(1)
C*
C*************************************************************************
C*                                                                        *
C*    START MARCHING DOWNSTREAM                                           *
C*                                                                        *
C*************************************************************************
C*
      DO 290 I = II , III
      WRITE(6,*)' '
      ASTR = 'CALCULATE THE FLOW FIELD FROM THE SECOND ORDER QUADRATIC WALL'
      CALL BANNER(ASTR)
      IF(I+1.GT.NII)THEN
```

```
       WRITE(6,*)' '
       WRITE(6,*)' Stopping the code: Variable NII needs to be increased.'
       STOP
      ENDIF

C*
C*********************************************************************
C*                                                                   *
C*    IF AT THE EXIT PLANE THEN END THE I LOOP                       *
C*                                                                   *
C*********************************************************************
C*
       IF(X(1).GE.XE)THEN
        EXIT_END=.TRUE.
        RETURN
       ENDIF
C*
C*********************************************************************
C*                                                                   *
C*    SET THE WALL CONDITIONS                                        *
C*                                                                   *
C*********************************************************************
C*
       L  = L + 1
       YW = Y(1)
       PW = P(1)
c      WRITE(9,*)' '
C*
C*********************************************************************
C*                                                                   *
C*    BEGIN LOOPING OVER THE J INDEX FROM THE WALL TO THE CENTERLINE   *
C*                                                                   *
C*********************************************************************
C*
 190   CONTINUE
       MFRT   = 0.0

       DO 270 J = 1 ,J2

        IF(J.EQ.1)THEN                                  !ON THE WALL
         CALL MOVE(2,2)                                 ! move lower point into P2
         CALL MOVE(3,1)                                 ! move wall point into P3
         CALL DIRECT_WALL_POINT

         IF(X4.GT.XE .AND. IVL.NE. 1)THEN               ! If past exit plane, reset X4
          X4 = XE                                       !  In general, this should not happen
          Y4 = YE                                       !  unless we reach the exit plane before
          A4=ATAN(LE)                                   !  we reach the design Mach number.
          L = L - 1
          CALL MOVE(3,1)
          CALL MOVE(1,2)
          L0 = LE                                      !slope at point P4
          CALL INVERSE_WALL_POINT
          J2 = J
          WRITE(9,*)' '
         ENDIF

         WRITE(6,1010)

        ELSEIF((J.EQ.J2).AND.(IEND.EQ.0))THEN           !ON THE AXIS
         CALL MOVE(1,J-1)                               ! MOVE the upper point TO P1
         CALL MOVE(6,0)                                 ! move the old P2 point from INTERIOR_POINT
into P3
         CALL AXIS

        ELSEIF(J.GT.J2)THEN
         WRITE(6,*)' Stopping Code - Error?'
         STOP
c        GOTO 290                                       !error condition, should not happen

        ELSE                                            !INTERIOR POINT
         CALL MOVE(1,J-1)                               ! MOVE the upper point TO P1
C...kdk
c        CALL MOVE(2,J+1)
         CALL MOVE(2,J)
         CALL INTERIOR_POINT

         IF(Y4.LT.0.0)THEN                               !if point is below axis call subroutine
AXIS
c         J2 = J
          CALL MOVE(1,J-1)                               ! MOVE the upper point TO P1
```

```
          CALL MOVE(6,0)                              ! move the old P2 point from
INTERIOR_POINT into P3
          CALL AXIS
         ELSE

c         IF(X4.LT.X(1))THEN
c          WRITE(6,*)' Stopping Code - Error?'
c          STOP
c          J2 = J2-J+1+IEND
c          L = L + J - 1
c          F = F - DF
c          DO K = 2 , J2
c           N = J - 1 + K
c           CALL MOVE(4,N)
c           CALL MOVE(5,K)
c          ENDDO
c          GOTO 190
c         ENDIF

         ENDIF

        ENDIF

        CALL MOVE(5,J)
C
C*************************************************************************
C*                                                                       *
C*    CALCULATE THE MASS FLOW RATE                                       *
C*                                                                       *
C*************************************************************************
C
       IF(J.EQ.1)THEN
        AREA_X = 0.0
        AREA_Y = 0.0
        DEL_X  = 0.0
        DEL_Y  = 0.0
        R_AVE  = 0.0
        U_AVE  = 0.0
        V_AVE  = 0.0
       ELSE
        DEL_X  = ABS(X4-X(J-1))
        DEL_Y  = ABS(Y4**2 - Y(J-1)**2 )*DELTA + (1-DELTA)*ABS(Y4-Y(J-1))
        AREA_X = PI * DEL_Y * DELTA + (1-DELTA)*DEL_Y
        AREA_Y = 2.0 * PI * (Y4 + Y(J-1))/2.0*DEL_X*DELTA + (1-DELTA)*DEL_X
        R_AVE  = (R4 + R(J-1))/2.0
        U_AVE  = (U4 + U(J-1))/2.0
        V_AVE  = (V4 + V(J-1))/2.0
       ENDIF

        MFR    = R_AVE*(U_AVE*AREA_X+V_AVE*AREA_Y)
        MFRT   = MFR + MFRT
        ERROR_M = (MFRT-MDOT)/MDOT

        AP = A4*RAD
        PP = P4/GL
        LJ = L + J

C*
C*************************************************************************
C*                                                                       *
C*    OUTPUT THE FLOW VARIABLES                                          *
C*                                                                       *
C*************************************************************************
C*
        IF (KW.EQ.1)THEN
         WRITE(6,1020)I,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
         WRITE(9,1020)I,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
         IF(J.EQ.1)WRITE(3,1020)I,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4

IF(J.EQ.1)WRITE(32,1020)I,J,X4,Y4,TM(T4),ZMUIFD(T4),RE(T4),BL(T4),Y4+BL_SCALE*0.1404*((1.+X4)**0.
125-1.0)
         IF(J.EQ.J2.AND.Y4.EQ.0.0.AND.X4.LE.XE)WRITE(4,1020)I,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
         IF(J.EQ.J2.AND.Y4.EQ.0.0.AND.X4.LE.XE)write(6,*)' MASS FLUX = ',MFRT,
     >    ' PERCENT ERROR = ',ERROR_M*100.
        ENDIF
C*
C*************************************************************************
C*                                                                       *
C*    PLACE THE FLOW VARIABLES INTO STORAGE                              *
C*                                                                       *
C*************************************************************************
```

```
C*
          XY(I,J,1)   = X4
          XY(I,J,2)   = Y4
          FLOW(I,J,1) = U4
          FLOW(I,J,2) = V4
          FLOW(I,J,3) = M4
          FLOW(I,J,4) = Q4
          FLOW(I,J,5) = A4
          FLOW(I,J,6) = P4
          FLOW(I,J,7) = R4
          FLOW(I,J,8) = T4
c         JMAX = MAX(J,JMAX)
C*
C***********************************************************************
C*                                                                     *
C*    CHECK FOR CROSSING LINES                                         *
C*                                                                     *
C***********************************************************************
C*
         IF(J.GE.2)THEN
          P1_X = XY(I,J,1)
          P1_Y = XY(I,J,2)
          P2_X = XY(I,J-1,1)
          P2_Y = XY(I,J-1,2)
          DO JJ = 2 , J2
           P3_X = XY(I-1,JJ,1)
           P3_Y = XY(I-1,JJ,2)
           P4_X = XY(I-1,JJ-1,1)
           P4_Y = XY(I-1,JJ-1,2)
           CALL LINE_INTERSECTION(P1_X,P1_Y,P2_X,P2_Y,P3_X,
     >     P3_Y,P4_X,P4_Y,H_FACTOR,G_FACTOR,IX,IY)
           IF(H_FACTOR.GE.0.0 .AND. H_FACTOR.LE.1.0 .AND.
     >        G_FACTOR.GE.0.0 .AND. G_FACTOR.LE.1.0 )THEN
            WRITE(6,*)' [31mCHARACTERISTICS INTERSECTION DEDECTED![0m'
C...interpolate for new point
            X4 = (1.0 - G_FACTOR)*P1_X + G_FACTOR*P2_X
            Y4 = (1.0 - G_FACTOR)*P1_Y + G_FACTOR*P2_Y
            U4 = (1.0 - G_FACTOR)*U4 + G_FACTOR*FLOW(I,J-1,1)
            V4 = (1.0 - G_FACTOR)*V4 + G_FACTOR*FLOW(I,J-1,2)
            M4 = (1.0 - G_FACTOR)*M4 + G_FACTOR*FLOW(I,J-1,3)
            Q4 = (1.0 - G_FACTOR)*Q4 + G_FACTOR*FLOW(I,J-1,4)
            A4 = (1.0 - G_FACTOR)*A4 + G_FACTOR*FLOW(I,J-1,5)
            P4 = (1.0 - G_FACTOR)*P4 + G_FACTOR*FLOW(I,J-1,6)
            R4 = (1.0 - G_FACTOR)*R4 + G_FACTOR*FLOW(I,J-1,7)
            T4 = (1.0 - G_FACTOR)*T4 + G_FACTOR*FLOW(I,J-1,8)
C*
C***********************************************************************
C*                                                                     *
C*    PLACE THE FLOW VARIABLES INTO STORAGE                            *
C*                                                                     *
C***********************************************************************
C*
            XY(I,J,1)   = X4
            XY(I,J,2)   = Y4
            FLOW(I,J,1) = U4
            FLOW(I,J,2) = V4
            FLOW(I,J,3) = M4
            FLOW(I,J,4) = Q4
            FLOW(I,J,5) = A4
            FLOW(I,J,6) = P4
            FLOW(I,J,7) = R4
            FLOW(I,J,8) = T4

            X(J) = X4
            Y(J) = Y4
            P(J) = P4
            R(J) = R4
            U(J) = U4
            V(J) = V4
            Q(J) = Q4
            A(J) = A4

            BACKSPACE(9)
            WRITE(9,1020)I,J,X4,Y4,U4,V4,M4,Q4,A4*RAD,P4/GL,R4,T4
            WRITE(6,1020)I,J,X4,Y4,U4,V4,M4,Q4,A4*RAD,P4/GL,R4,T4
c           WRITE(9,1020)I,JJJ,X4,Y4,U4,V4,M4,Q4,A4*RAD,P4/GL,R4,T4
c           WRITE(6,1020)I,JJJ,X4,Y4,U4,V4,M4,Q4,A4*RAD,P4/GL,R4,T4

            DO JJJ = J+1 , J2
             XY(I,JJJ,1)   = XY(I-1,JJJ,1)
             XY(I,JJJ,2)   = XY(I-1,JJJ,2)
```

```
            FLOW(I,JJJ,1) = FLOW(I-1,JJJ,1)
            FLOW(I,JJJ,2) = FLOW(I-1,JJJ,2)
            FLOW(I,JJJ,3) = FLOW(I-1,JJJ,3)
            FLOW(I,JJJ,4) = FLOW(I-1,JJJ,4)
            FLOW(I,JJJ,5) = FLOW(I-1,JJJ,5)
            FLOW(I,JJJ,6) = FLOW(I-1,JJJ,6)
            FLOW(I,JJJ,7) = FLOW(I-1,JJJ,7)
            FLOW(I,JJJ,8) = FLOW(I-1,JJJ,8)
            X(JJJ) = XY(I,JJJ,1)
            Y(JJJ) = XY(I,JJJ,2)
            P(JJJ) = FLOW(I,JJJ,6)
            R(JJJ) = FLOW(I,JJJ,7)
            U(JJJ) = FLOW(I,JJJ,1)
            V(JJJ) = FLOW(I,JJJ,2)
            Q(JJJ) = FLOW(I,JJJ,4)
            A(JJJ) = FLOW(I,JJJ,5)
            LJ = L + JJJ
            WRITE(6,1020)I,JJJ,XY(I,JJJ,1),XY(I,JJJ,2),FLOW(I,JJJ,1),FLOW(I,JJJ,2),
     >
FLOW(I,JJJ,3),FLOW(I,JJJ,4),FLOW(I,JJJ,5)*RAD,FLOW(I,JJJ,6)/GL,FLOW(I,JJJ,7),FLOW(I,JJJ,8)
            WRITE(9,1020)I,JJJ,XY(I,JJJ,1),XY(I,JJJ,2),FLOW(I,JJJ,1),FLOW(I,JJJ,2),
     >
FLOW(I,JJJ,3),FLOW(I,JJJ,4),FLOW(I,JJJ,5)*RAD,FLOW(I,JJJ,6)/GL,FLOW(I,JJJ,7),FLOW(I,JJJ,8)
           ENDDO
C
C***********************************************************************
C*                                                                     *
C*     SMOOTH THE PROFILE                                              *
C*                                                                     *
C***********************************************************************
C
           IF(SMOOTH)THEN
            DO LL = 1 , J2
             BACKSPACE(9)
            ENDDO
            J_START = 1
            J_END   = J2
            I_LINE = I
            I_FILE = 9
            CALL SMOOTH_PROFILE(J_START,J_END,I_LINE,0,I_FILE)
           ENDIF

          GOTO 290

          ENDIF

         ENDDO
C*
C***********************************************************************
C*                                                                     *
C*     END OF LINE CROSSING CHECK                                      *
C*                                                                     *
C***********************************************************************
C*
        ENDIF
C*
C***********************************************************************
C*                                                                     *
C*     AT THE END OF EACH J INDEX LOOP WE NEED TO CHECK IF THE         *
C*      CENTERLINE MACH NUMBER HAS REACHED THE DESIGNED MACH NUMBER.    *
C*                                                                     *
C***********************************************************************
C*
      IF(NOZ.EQ.0 .AND. Y4.EQ.0.0 .AND. M4.GE.EMD)THEN

       J_START  =  1
       J_END    = J2
       L_OFFSET =  L
       L_OFFSET =  0
       I_LINE   =  I

C... need to backspace the plot files

c      BACKSPACE(3)
       write(6,*)'I_LINE,J_END =',I_LINE,J_END

       DO LL = 1 , J2
        BACKSPACE(9)
       ENDDO

c      DO LL = 1 , J2
```

```
c        WRITE(9,*)' '
c       ENDDO
C*
C***********************************************************************
C*                                                                    *
C*    FIND THE LOCATION WHERE MACH NUMBER = DESIGNED MACH NUMBER      *
C*                                                                    *
C***********************************************************************
C*
       CALL MLINE(J_START,J_END,I_LINE,L_OFFSET,9)

       X4 = XY(I,J_END,1)
       Y4 = XY(I,J_END,2)
       U4 = FLOW(I,J_END,1)
       V4 = FLOW(I,J_END,2)
       M4 = FLOW(I,J_END,3)
       Q4 = FLOW(I,J_END,4)
       A4 = FLOW(I,J_END,5)
       P4 = FLOW(I,J_END,6)
       R4 = FLOW(I,J_END,7)
       T4 = FLOW(I,J_END,8)

       DESIGNED_M=.TRUE.

       RETURN

      ENDIF
C*
C***********************************************************************
C*                                                                    *
C*    CHECK FOR EXIT PLANE LOCATION                                   *
C*                                                                    *
C***********************************************************************
C*
        IMAX = I

        IF( J.EQ.1 .AND. X4 .GE. XE .AND. IVL.NE.1)THEN
         EXIT_END = .TRUE.
         RETURN
        ENDIF
C
C***********************************************************************
C*                                                                    *
C*    END OF J INDEX LOOP                                             *
C*                                                                    *
C***********************************************************************
C
 270   CONTINUE
C
C***********************************************************************
C*                                                                    *
C*    SMOOTH THE PROFILE                                              *
C*                                                                    *
C***********************************************************************
C
      IF(SMOOTH)THEN
       DO LL = 1 , J2
        BACKSPACE(9)
       ENDDO
       J_START  =  1
       J_END    = J2
       L_OFFSET =  L
       I_LINE   =  I
       I_FILE   =  9
       CALL SMOOTH_PROFILE(J_START,J_END,I_LINE,L_OFFSET,I_FILE)
      ENDIF

c     IF(IEND.EQ.1) J2 = J2 - 1
c     GOTO 290

c280  J2 = J - 1
c     IEND = 1
C
C***********************************************************************
C*                                                                    *
C*    END OF I INDEX LOOP                                             *
C*                                                                    *
C***********************************************************************
C
 290  CONTINUE
C
```

```
C*********************************************************************
C*                                                                   *
C*    SET THE ENDING I INDEX TO BE USED LATER                        *
C*                                                                   *
C*********************************************************************
C
      IMAX = I
C
C*********************************************************************
C*                                                                   *
C*    RETURN TO THE MAIN CALLING PROGRAM                             *
C*                                                                   *
C*********************************************************************
C
      RETURN
C
C*********************************************************************
C*                                                                   *
C*    FORMAT STATEMENTS                                              *
C*                                                                   *
C*********************************************************************
C
 1010 FORMAT(/,'    I    J ',','   X        ', '     Y      ',' U VELOCITY ', ' V VELOCITY ',
     >'MACH NUMBER ', 'VELOCITY MAG',' FLOW ANGLE ',' PRESSURE   ','  DENSITY   ',' TEMPERATURE')
 1020 FORMAT(2I5,1P10E12.4)
C
C*********************************************************************
C*                                                                   *
C*    END OF LINE                                                    *
C*                                                                   *
C*********************************************************************
C
      END
      SUBROUTINE HUNT(XX,N,X,JLO)
C
C********************************************************************
C**                                                                **
C**   HUNT FOR NEAREST NEIGHBOR IN XX FOR X                        **
C**                                                                **
C**   INPUT:                                                       **
C**   XX(MAXD)      ARRAY OF OLD X DATA                            **
C**   N             MAXIMUM DIAMENSION OF XX                       **
C**   X             X POINT AT WHICH XX IS TO BE SCANED FOR        **
C**                                                                **
C**   OUTPUT:                                                      **
C**   JLO           NEW INDEX VALUE WHERE XX(JLO) < X < XX(JLO+1)  **
C**                                                                **
C********************************************************************
C
C
C********************************************************************
C**                                                                **
C**   IMPLICIT STATEMENTS                                          **
C**                                                                **
C********************************************************************
C
      INTEGER JLO,N
      REAL X,XX(N)
      INTEGER INC,JHI,JM
      LOGICAL ASCND
C
C********************************************************************
C**                                                                **
C**   TRUE IF ASCENDING ORDER                                      **
C**                                                                **
C********************************************************************
C
      ASCND=XX(N).GT.XX(1)

      IF(JLO.LE.0.OR.JLO.GT.N)THEN
        JLO=0
        JHI=N+1
        GOTO 3
      ENDIF

      INC=1

      IF(X.GE.XX(JLO).EQV.ASCND)THEN
1       JHI=JLO+INC
        IF(JHI.GT.N)THEN
          JHI=N+1
```

```
          ELSE IF(X.GE.XX(JHI).EQV.ASCND)THEN
            JLO=JHI
            INC=INC+INC
            GOTO 1
          ENDIF
        ELSE
          JHI=JLO
2         JLO=JHI-INC
          IF(JLO.LT.1)THEN
            JLO=0
          ELSE IF(X.LT.XX(JLO).EQV.ASCND)THEN
            JHI=JLO
            INC=INC+INC
            GOTO 2
          ENDIF
        ENDIF

3       IF(JHI-JLO.EQ.1)RETURN

        JM=(JHI+JLO)/2

        IF(X.GT.XX(JM).EQV.ASCND)THEN
          JLO=JM
        ELSE
          JHI=JM
        ENDIF

        GOTO 3

        END
      SUBROUTINE SMOOTH_PROFILE(J_START,J_END,I_LINE,L_OFFSET,I_FILE)
C
C*********************************************************************
C**                                                                 **
C**   SUBROUTINE SMOOTH_PROFILE WILL SPREAD OUT THE POINTS ALONG    **
C**   ALONG THE MARCHING FRONT TO PREVENT POINTS FROM GATHERING     **
C**   TOGETHER. THIS HELPS WEAK WAVES FROM BECOMING STRONGER.       **
C**                                                                 **
C**   WHERE:                                                        **
C**    J_START  = STARTING LOCATION ALONG THE WAVE FRONT            **
C**    J_END    = ENDING   LOCATION ALONG THE WAVE FRONT            **
C**    I_LINE   = I INDEX OF THE WAVE FRONT                         **
C**    L_OFFSET = OUTPUT OFF SET ON J INDEX                         **
C**                                                                 **
C*********************************************************************
C
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'

      DEBUG = .FALSE.
C
C**********************************************************************
C*                                                                    *
C*    OUTPUT THE HEADER STATEMENTS                                    *
C*                                                                    *
C**********************************************************************
C
       WRITE(6,*)' '
       ASTR = 'CORRECTING PROFILE: SMOOTHING STEP'
       CALL BANNER(ASTR)
C
C**********************************************************************
C*                                                                    *
C*    SET INITIAL CONSTANTS                                           *
C*                                                                    *
C**********************************************************************
C
      MFRT    = 0.0
      S       = 0.0                      !increment path lenght
      STP     = 0.0                      !path length
      J_DELTA = J_END - J_START + 1      !Number of points
      I_LINE  = I_LINE + 1               !put new data into the next I line

      IF(DEBUG)write(6,*)' Number of points =',J_DELTA
C
C**********************************************************************
C*                                                                    *
C*    FIND THE TOTAL LINEAR PATH LENGTH OF THE WAVE FRONT             *
C*                                                                    *
C**********************************************************************
C
```

```
      DO  J = J_START, J_END

c       IF(J.NE.J_START)S = SQRT((Y(J) - Y(J-1))**2
c     >                        + (X(J) - X(J-1))**2) !find increment path length
        IF(J.NE.J_START)S = SQRT((XY(I_LINE-1,J,2) - XY(I_LINE-1,J-1,2))**2
     >                        + (XY(I_LINE-1,J,1) - XY(I_LINE-1,J-1,1))**2) !find increment path
length

        IF(DEBUG)write(6,*)'J,S=',J,S
        STP = S + STP   !total path length

      ENDDO
C
C**********************************************************************
C*                                                                    *
C*    FIND THE STEP SIZE ON NEW PATH                                  *
C*                                                                    *
C**********************************************************************
C
      DS = STP / (J_DELTA*1.-1.)      !break up path in equal potions

      IF(DEBUG)write(6,*)' Total path length =',STP
      IF(DEBUG) WRITE(6,*)' DS = ',DS

      WRITE(6,1010)
      WRITE(I_FILE,*)' '

      IF(DEBUG)write(6,*)' Number of points =',J_DELTA
C
C**********************************************************************
C*                                                                    *
C*    SET THE PATH LENGTH TO ZERO                                     *
C*                                                                    *
C**********************************************************************
C
      S = 0
C
C**********************************************************************
C*                                                                    *
C*    START SPACING THE DATA BY LOOPING OVER J                        *
C*                                                                    *
C**********************************************************************
C
      DO  J = J_START, J_END
C
C**********************************************************************
C*                                                                    *
C*    INCREMENT THE PATH LENGTH                                       *
C*                                                                    *
C**********************************************************************
C
       IF(J.NE.J_START)S = S + DS
C
C**********************************************************************
C*                                                                    *
C*    FIX THE STARTING POINT                                          *
C*                                                                    *
C**********************************************************************
C
       IF(J.EQ.J_START)THEN
        ZETA = 0.0
        J1   = J_START+1
        S    = 0.0
        GOTO 10
       ENDIF
C
C**********************************************************************
C*                                                                    *
C*    FIX THE ENDING POINT                                            *
C*                                                                    *
C**********************************************************************
C
       IF(J.EQ.J_END)THEN
        ZETA = 1.0
        J1   = J_END
        S    = STP
        GOTO 10
       ENDIF

       ST1_OLD = STP
       ST1     = 0.0
```

```
C
C*********************************************************************
C*                                                                   *
C*    FIND THE (X,Y) POINT ON PATH S                                 *
C*                                                                   *
C*********************************************************************
C
       DO J1 = J_START+1, J_END

        S1= SQRT((Y(J1) - Y(J1-1))**2 + (X(J1) - X(J1-1))**2)
        ST1 = S1 + ST1
C
C*********************************************************************
C*                                                                   *
C*    SEARCH FOR THE LOCAL PATH GREATER OR EQUAL TO PATH S           *
C*                                                                   *
C*********************************************************************
C
        IF(ST1.GE.S)THEN
         ZETA      = (S-ST1_OLD) / (ST1 - ST1_OLD)
c      write(6,*)'ZETA,S,ST1,ST1_OLD,J1',ZETA,S,ST1,ST1_OLD,J1
         GOTO 10
        ENDIF

        ST1_OLD = ST1
C
C*********************************************************************
C*                                                                   *
C*    END OF THE SEARCH LOOP J                                       *
C*                                                                   *
C*********************************************************************
C
       ENDDO

       ZETA = 1.0

 10    CONTINUE
C
C*********************************************************************
C*                                                                   *
C*    SINCE WE ADVANCED BY ONE POINT RESET JLO BACK BY ONE           *
C*                                                                   *
C*********************************************************************
C
       JLO = J1-1
C
C*********************************************************************
C*                                                                   *
C*    FIND THE NEW VALUE BASE ON ZETA                                *
C*                                                                   *
C*********************************************************************
C
       XY(I_LINE,J,1) = (1.-ZETA)*X(JLO) + ZETA*X(JLO+1)
       XY(I_LINE,J,2) = (1.-ZETA)*Y(JLO) + ZETA*Y(JLO+1)
       DO N = 1 , 8
        FLOW(I_LINE,J,N)   = (1.-ZETA)*FLOW(I_LINE-1,JLO,N) + ZETA*FLOW(I_LINE-1,JLO+1,N)
       ENDDO
c      WRITE(6,*)'X,YY,JLO,ZETA=',XY(I_LINE,J,1),YY,JLO,ZETA
      ENDDO
C
C*********************************************************************
C*                                                                   *
C*    RELOAD THE NEW DATA INTO THE ARRAY                             *
C*                                                                   *
C*********************************************************************
C
      DO  J = J_START , J_END

      X(J) = XY(I_LINE,J,1)
      Y(J) = XY(I_LINE,J,2)
      P(J) = FLOW(I_LINE,J,6)
      R(J) = FLOW(I_LINE,J,7)
      U(J) = FLOW(I_LINE,J,1)
      V(J) = FLOW(I_LINE,J,2)
c     Q(J) = FLOW(I_LINE,J,4)
      Q(J) = SQRT(FLOW(I_LINE,J,1)**2 + FLOW(I_LINE,J,2)**2)
c     A(J) = FLOW(I_LINE,J,5)
      A(J) = ATAN(FLOW(I_LINE,J,2)/FLOW(I_LINE,J,1))
      X4 = X(J)
      Y4 = Y(J)
      P4 = P(J)
```

```
      R4 = R(J)
      Q4 = Q(J)
      A4 = A(J)
      U4 = U(J)
      V4 = V(J)
      CALL THERMO(Q4,P4,R4,T4,C,M4)

      XY(I_LINE-1,J,1)   = X(J)
      XY(I_LINE-1,J,2)   = Y(J)
      FLOW(I_LINE-1,J,1) = U(J)
      FLOW(I_LINE-1,J,2) = V(J)
      FLOW(I_LINE-1,J,3) = M4
      FLOW(I_LINE-1,J,4) = Q(J)
      FLOW(I_LINE-1,J,5) = A(J)
      FLOW(I_LINE-1,J,6) = P(J)
      FLOW(I_LINE-1,J,7) = R(J)
      FLOW(I_LINE-1,J,8) = T4
C
C***********************************************************************
C*                                                                     *
C*    FIND THE MASS FLUX                                               *
C*                                                                     *
C***********************************************************************
C
       IF(J.EQ.J_START)THEN
        AREA_X = 0.0
        AREA_Y = 0.0
        DEL_X  = 0.0
        DEL_Y  = 0.0
        R_AVE  = 0.0
        U_AVE  = 0.0
        V_AVE  = 0.0
       ELSE
        DEL_X  = ABS(X4-X(J-1))
        DEL_Y  = ABS((Y4**2 - Y(J-1)**2 ))*DELTA + (1-DELTA)*ABS(Y4-Y(J-1))
        AREA_X = PI * DEL_Y * DELTA + (1-DELTA)*DEL_Y
        AREA_Y = 2.0 * PI * (Y4 + Y(J-1))/2.0*DEL_X*DELTA + (1-DELTA)*DEL_X
        R_AVE  = (R4 + R(J-1))/2.0
        U_AVE  = (U4 + U(J-1))/2.0
        V_AVE  = (V4 + V(J-1))/2.0
       ENDIF

        MFR    = R_AVE*(U_AVE*AREA_X+V_AVE*AREA_Y)
c       WRITE(6,*)'J,MFR,R_AVE=',J,MFR,R_AVE
        MFRT   = MFR + MFRT
        ERROR_M = (MFRT-MDOT)/MDOT

      ENDDO
C
C***********************************************************************
C*                                                                     *
C*    DUE TO THE LINEAR INTERPOLATION WE NEED TO FIX THE MASS FLUX     *
C*     BY CORRECTING THE DENSITY                                       *
C*                                                                     *
C***********************************************************************
C
      CORRECTION = MDOT/MFRT
      DO  J = J_START , J_END
       R(J) = FLOW(I_LINE,J,7)*MDOT/MFRT
       R4   = R(J)
       FLOW(I_LINE-1,J,7) = R(J)
      ENDDO
      MFRT = 0.0
      DO  J = J_START , J_END
      X4 = X(J)
      Y4 = Y(J)
      P4 = P(J)
      R4 = R(J)
      Q4 = Q(J)
      A4 = A(J)
      U4 = U(J)
      V4 = V(J)
      CALL THERMO(Q4,P4,R4,T4,C,M4)
       IF(J.EQ.J_START)THEN
        AREA_X = 0.0
        AREA_Y = 0.0
        DEL_X  = 0.0
        DEL_Y  = 0.0
        R_AVE  = 0.0
        U_AVE  = 0.0
        V_AVE  = 0.0
```

```
       ELSE
        DEL_X  = ABS(X4-X(J-1))
        DEL_Y  = ABS((Y4**2 - Y(J-1)**2 ))*DELTA + (1-DELTA)*ABS(Y4-Y(J-1))
        AREA_X = PI * DEL_Y * DELTA + (1-DELTA)*DEL_Y
        AREA_Y = 2.0 * PI * (Y4 + Y(J-1))/2.0*DEL_X*DELTA + (1-DELTA)*DEL_X
        R_AVE  = (R4 + R(J-1))/2.0
        U_AVE  = (U4 + U(J-1))/2.0
        V_AVE  = (V4 + V(J-1))/2.0
       ENDIF

        MFR    = R_AVE*(U_AVE*AREA_X+V_AVE*AREA_Y)
c       WRITE(6,*)'J,MFR,R_AVE=',J,MFR,R_AVE
        MFRT   = MFR + MFRT
        ERROR_M = (MFRT-MDOT)/MDOT
      ENDDO
C
C*********************************************************************
C*                                                                   *
C*    OUTPUT THE NEW SOLUTION                                        *
C*                                                                   *
C*********************************************************************
C
      DO  J = J_START , J_END
      X4 = X(J)
      Y4 = Y(J)
      P4 = P(J)
      R4 = R(J)
      Q4 = Q(J)
      A4 = A(J)
      U4 = U(J)
      V4 = V(J)
      CALL THERMO(Q4,P4,R4,T4,C,M4)
        AP = A4*RAD
        PP = P4 /GL

       IF(J.EQ.1)THEN
        BACKSPACE(3)
        WRITE(3,1020)I_LINE,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
       ENDIF

       WRITE(6,1020)I_LINE-1,J+L_OFFSET,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
       WRITE(I_FILE,1020)I_LINE-1,J+L_OFFSET,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
       IF(J.EQ.J_END)write(6,*)' MASS FLUX = ',MFRT,
     >   ' PERCENT ERROR = ',ERROR_M*100., CORRECTION

      ENDDO

      I_LINE     = I_LINE - 1
c     STOP
C
C*********************************************************************
C*                                                                   *
C*    FORMAT STATEMENTS                                              *
C*                                                                   *
C*********************************************************************
C
 1010 FORMAT(/,'    I    J ','   X        ', '     Y      ',' U VELOCITY ', ' V VELOCITY ',
     >'MACH NUMBER ', 'VELOCITY MAG',' FLOW ANGLE ',' PRESSURE   ','  DENSITY   ',' TEMPERATURE')
 1020 FORMAT(2I5,1P10E12.4)
C
C*********************************************************************
C*                                                                   *
C*    RETURN STATEMENT                                               *
C*                                                                   *
C*********************************************************************
C
      RETURN
C
C*********************************************************************
C*                                                                   *
C*    END OF LINE                                                    *
C*                                                                   *
C*********************************************************************
C
      END
      SUBROUTINE INTERIOR_POINT
C...
C...  This subroutine computes the interior region.
C...
C...  Points 1 and 2 are known
C...  Point 4 is the intersection of points 1 and 2
```

```
C...  Point 3 is located between points 1 and 2
C...  The C- characteristic goes through points 1 and 4
C...  The C+ characteristic goes through points 2 and 4
C...  The streamline        goes through points 3 and 4
C...
C...   1*
C...     *  C-
C...      *
C...       *
C...   3***** 4
C...       *
C...      *
C...     * C+
C...   2*

C...
C... Calculate the solution at an interior point
C...
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C...
C... CALCULATE THE COEFFICIENTS FOR THE PREDICTOR
C...
      ITER = 0

      IF(X1-X2.NE.0)THEN
       L12 = (Y1-Y2)/(X1-X2)
      ELSE
       L12 = 0.0
      ENDIF

      CALL THERMO (Q2,P2,R2,T,C,M)

      LP = TAN(A2+ASIN(1./M))
      QP = GC*SQRT(M**2-1.)/(R2*Q2**2)
      SP = DELTA/(M*COS(A2+ASIN(1./M)))

      IF(Y2.EQ.0.0) SP = SP * SIN(A1)/Y1
      IF(Y2.GT.0.0) SP = SP * SIN(A2)/Y2

      CALL THERMO (Q1,P1,R1,T,C,M)

      LM = TAN(A1-ASIN(1./M))
      QM = GC*SQRT(M**2-1.)/(R1*Q1**2)
      SM = DELTA*SIN(A1)/(Y1*M*COS(A1-ASIN(1./M)))
      A3 = 0.5*(A1+A2)
      A4 = A3
C...
C... PROJECT CHARACTERSITIC LINE AND FIND UNKNOWN POINT 4
C...
 10   X4 = (Y1 -Y2-lm*x1+lp*x2)/(lp-lm)
      Y4 = Y1+LM*(X4-X1)

      IF(Y4.LT.0.0)RETURN

      TP = -SP*(X4-X2)+QP*P2+A2
      TM = -SM*(X4-X1)+QM*P1-A1

      K  = 1
C...
C... INTERPOLATE BACK FROM POINT 4 ALONG THE STREAM LINE TO FIND POINT 3
C...  (THE POINT BETWEEN 1 AND 2)
C...
 20   L0 = TAN(0.5*(A3+A4))
      IF(L12.NE.0.0)THEN
      X3 = (Y4-Y2-L0*X4+L12*X2)/(L12-L0)
      ELSE
      X3 = X1
      ENDIF
      Y3 = Y4+L0*(X3-X4)

      D  = (Y3-Y2)/(Y1-Y2)

      A3 = A2+D*(A1-A2)

      IF(ITER.EQ.0)A4=A3
      IF(K.GT.1 .AND. ABS(Y3-YC).LT.ERROR) GOTO 30
C...ERROR in decode
      IF(K.GT.1000)THEN
       WRITE(6,*)' ERROR: CAN NOT DECODE FLOW ANGLE!'
       WRITE(6,*)'Y3 =',Y3
```

```
       WRITE(6,*)'YC =',YC
       WRITE(6,*)'Y3-YC =',Y3-YC
       WRITE(6,*)' '
 555   FORMAT(1P4E15.5)
       WRITE(6,*)'X = ',X1,X2,X3
       WRITE(6,*)'Y = ',Y1,Y2,Y3
       WRITE(6,*)'Q = ',Q1,Q2,Q3
       WRITE(6,*)'P = ',P1,P2,P3
       WRITE(6,*)'R = ',R1,R2,R3
       WRITE(6,*)'A = ',A1,A2,A3
       STOP
      ENDIF

      XC = X3
      YC = Y3

      K  = K + 1
      GOTO 20

 30   Q3 = Q2+D*(Q1-Q2)
      P3 = P2+D*(P1-P2)
      R3 = R2+D*(R1-R2)

      IF(ITER.GT.0)GOTO 40

      Q4 = Q3
      P4 = P3
      R4 = R3

 40   P0 = 0.5*(P3+P4)
      R8 = 0.5*(R3+R4)
      Q0 = 0.5*(Q3+Q4)
      R0 = R8*Q0/GC

      CALL THERMO (Q0,P0,R8,T,C,M)

      A0 = C**2/GC
      T01=R0*Q3+P3
      T02=P3-A0*R3
C...
C... CALCULATE THE PROPERITIES AT POINT 4
C...
      P4 = (TP+TM)/(QP+QM)
      A4 = TP-QP*P4
      Q4 = (T01-P4)/R0
      R4 = (P4-T02)/A0
C...
C... TEST FOR CONVERGENCE
C...
c     WRITE(6,*)' '
c     WRITE(6,100)' R0                 = ',R0
c     WRITE(6,100)' A0                 = ',A0
c     WRITE(6,100)' T01                = ',T01
c     WRITE(6,100)' T02                = ',T02
c     WRITE(6,100)' Q+                 = ',QP
c     WRITE(6,100)' S+                 = ',SP
c     WRITE(6,100)' T+                 = ',TP
c     WRITE(6,100)' Q-                 = ',QM
c     WRITE(6,100)' S-                 = ',SM
c     WRITE(6,100)' T-                 = ',TM
      IF(ITER.EQ.ICOR) RETURN
      IF(ITER.EQ.0)    GOTO 50

      IF((ABS(X4-XD).GT.E1)    .OR.  (ABS(Y4-YD).GT.E1))    GOTO 50
      IF((ABS(P4-PD).GT.E2*PD) .OR.  (ABS(R4-RD).GT.E3*RD)) GOTO 50
      IF((ABS(Q4-QD).GT.E4*QD) .AND. (ABS(A4-AD).GT.E5*AD)) RETURN
C...
C... CALCULATE THE COEFFICIENTS FOR THE CORRECTOR
C...
 50   ITER = ITER + 1

      XD = X4
      YD = Y4
      PD = P4
      RD = R4
      QD = Q4
      AD = A4

      P0  = 0.5*(P2+P4)
      R0  = 0.5*(R2+R4)
      Q0  = 0.5*(Q2+Q4)
```

```
      A8  = 0.5*(A2+A4)
      Y0  = 0.5*(Y2+Y4)

      CALL THERMO (Q0,P0,R0,T,C,M)

      LP = TAN(A8+ASIN(1.0/M))
      QP = GC*SQRT(M**2-1.0)/(R0*Q0**2)

      P0  = 0.5*(P1+P4)
      R0  = 0.5*(R1+R4)

      SP = DELTA*SIN(A8)/(Y0*M*COS(A8+ASIN(1.0/M)))

      Q0  = 0.5*(Q1+Q4)
      A8  = 0.5*(A1+A4)
      Y0  = 0.5*(Y1+Y4)

      CALL THERMO (Q0,P0,R0,T,C,M)

      LM = TAN(A8-ASIN(1.0/M))
      QM = GC*SQRT(M**2-1.0)/(R0*Q0**2)

      SM = DELTA*SIN(A8)/(Y0*M*COS(A8-ASIN(1.0/M)))

 100  FORMAT(A,1P2E15.5)
      GOTO 10

      END
      SUBROUTINE MLINE(J_START,J_END,I_LINE,L_OFFSET,I_FILE)
C
C***********************************************************************
C*                                                                     *
C*    FIND NEW MACH LINE                                               *
C*                                                                     *
C***********************************************************************
C
C*
C***********************************************************************
C*                                                                     *
C*    INCLUDE AND COMMON STATEMENTS                                    *
C*                                                                     *
C***********************************************************************
C*
      INCLUDE 'MOC.PAR'
      INCLUDE 'MOC.CMN'
C
C***********************************************************************
C*                                                                     *
C*    OUTPUT THE HEADER                                                *
C*                                                                     *
C***********************************************************************
C
      WRITE(6,*)' '
      ASTR = 'INTERPOLATE TO FIND NEW LOCATION AT THE DESIGNED MACH NUMBER'
      CALL BANNER(ASTR)

      WRITE(6,1010)
c     WRITE(I_FILE,*)' '

c     DO JI = J_START,J_END
c      IF(XY(I_LINE-1,JI,2).EQ.0.0 .AND. XY(I_LINE-1,JI,1).NE.0.0)THEN
c      ENDIF
c     ENDDO

      JI = J_END - 1

c     write(6,*)'FLOW(I_LINE-1,JI,3)=',FLOW(I_LINE-1,JI,3)
c     write(6,*)'FLOW(I_LINE,J_END,3)=',FLOW(I_LINE,J_END,3)

      ZETA = (EMD - FLOW(I_LINE-1,J_END,3)) / (FLOW(I_LINE,J_END,3) - FLOW(I_LINE-1,J_END,3))

c     WRITE(6,*)'ZETA=',ZETA
       IF(ZETA.LT.0.0)THEN
        WRITE(6,*)'Warning: ZETA=',ZETA
        ZETA = 0.0
       ENDIF
       IF(ZETA.GT.1.0)THEN
        WRITE(6,*)'Warning: ZETA=',ZETA
        ZETA = 1.0
       ENDIF
C...interior points
```

```
      DO  J = J_END-1, J_START+1, -1
       XY(I_LINE,J,1) = (1.-ZETA)*XY(I_LINE-1,J,1) + ZETA*XY(I_LINE,J-1,1)
       XY(I_LINE,J,2) = (1.-ZETA)*XY(I_LINE-1,J,2) + ZETA*XY(I_LINE,J-1,2)
       DO N = 1 , 8
        FLOW(I_LINE,J,N)   = (1.-ZETA)*FLOW(I_LINE-1,J,N) + ZETA*FLOW(I_LINE,J-1,N)
       ENDDO
      ENDDO
C...Wall point
       XY(I_LINE,1,1) = (1.-ZETA)*XY(I_LINE-1,1,1) + ZETA*XY(I_LINE,1,1)
       XY(I_LINE,1,2) = (1.-ZETA)*XY(I_LINE-1,1,2) + ZETA*XY(I_LINE,1,2)
       DO N = 1 , 8
        FLOW(I_LINE,1,N)   = (1.-ZETA)*FLOW(I_LINE-1,1,N) + ZETA*FLOW(I_LINE,1,N)
       ENDDO
C...Centerline point
      J = J_END
       XY(I_LINE,J,1)    = (1.-ZETA)*XY(I_LINE-1,J,1) + ZETA*XY(I_LINE,J_END,1)
       XY(I_LINE,J,2)    = (1.-ZETA)*XY(I_LINE-1,J,2) + ZETA*XY(I_LINE,J_END,2)
       DO N = 1 , 8
        FLOW(I_LINE,J,N) = (1.-ZETA)*FLOW(I_LINE-1,J,N) + ZETA*FLOW(I_LINE,J_END,N)
       ENDDO

C...Reload the new data
      DO  J = J_START , J_END

      X(J) = XY(I_LINE,J,1)
      Y(J) = XY(I_LINE,J,2)
      P(J) = FLOW(I_LINE,J,6)
      R(J) = FLOW(I_LINE,J,7)
      U(J) = FLOW(I_LINE,J,1)
      V(J) = FLOW(I_LINE,J,2)
c     Q(J) = FLOW(I_LINE,J,4)
      Q(J) = SQRT(FLOW(I_LINE,J,1)**2 + FLOW(I_LINE,J,2)**2)
c     A(J) = FLOW(I_LINE,J,5)
      A(J) = ATAN(FLOW(I_LINE,J,2)/FLOW(I_LINE,J,1))

      X4 = X(J)
      Y4 = Y(J)
      P4 = P(J)
      R4 = R(J)
      Q4 = Q(J)
      A4 = A(J)
      U4 = U(J)
      V4 = V(J)
      CALL THERMO(Q4,P4,R4,T4,C,M4)

      FLOW(I_LINE,J,1) = U(J)
      FLOW(I_LINE,J,2) = V(J)
      FLOW(I_LINE,J,3) = M4
      FLOW(I_LINE,J,4) = Q(J)
      FLOW(I_LINE,J,5) = A(J)
      FLOW(I_LINE,J,6) = P(J)
      FLOW(I_LINE,J,7) = R(J)
      FLOW(I_LINE,J,8) = T4

      ENDDO

C...Output the new data

      MFRT = 0.0
      DO  J = J_START , J_END
      X4 = X(J)
      Y4 = Y(J)
      P4 = P(J)
      R4 = R(J)
      Q4 = Q(J)
      A4 = A(J)
      U4 = U(J)
      V4 = V(J)
      CALL THERMO(Q4,P4,R4,T4,C,M4)

      AP = A4*RAD
      PP = P4 /GL

       IF(J.EQ.1)THEN
        AREA_X = 0.0
        AREA_Y = 0.0
        DEL_X  = 0.0
        DEL_Y  = 0.0
        R_AVE  = 0.0
        U_AVE  = 0.0
        V_AVE  = 0.0
```

```
       ELSE
        DEL_X  = ABS(X4-X(J-1))
        DEL_Y  = ABS(Y4-Y(J-1))
        AREA_X = PI * ABS(Y4**2 - Y(J-1)**2 ) * DELTA + (1-DELTA)*ABS(Y4-Y(J-1))
        AREA_Y = 2.0 * PI * (Y4 + Y(J-1))/2.0*DEL_X* DELTA + (1-DELTA)*DEL_X
        R_AVE  = (R4 + R(J-1))/2.0
        U_AVE  = (U4 + U(J-1))/2.0
        V_AVE  = (V4 + V(J-1))/2.0
       ENDIF

        MFR    = R_AVE*(U_AVE*AREA_X+V_AVE*AREA_Y)
        MFRT   = MFR + MFRT
        ERROR_M = (MFRT-MDOT)/MDOT

      IF (KW.EQ.1)THEN
       IF(J.EQ.1)THEN
        BACKSPACE(3)
        WRITE(3,1020)I_LINE,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
       ENDIF
       WRITE(6,1020)I_LINE,J+L_OFFSET,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
c      WRITE(I_FILE,1020)I_LINE,J+L_OFFSET,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
       IF(J.EQ.J_END)write(6,*)' MASS FLUX = ',MFRT,
     >   ' PERCENT ERROR = ',ERROR_M*100.
      ENDIF

      ENDDO

 1010 FORMAT(/,'    I    J ','   X        ', '     Y      ',' U VELOCITY ', ' V VELOCITY ',
     >'MACH NUMBER ', 'VELOCITY MAG',' FLOW ANGLE ',' PRESSURE   ','  DENSITY   ',' TEMPERATURE')
 1020 FORMAT(2I5,1P10E12.4)

c     GOTO 30

C
C*********************************************************************
C*                                                                   *
C*    OUTPUT THE HEADER                                              *
C*                                                                   *
C*********************************************************************
C
      WRITE(6,*)' '
      ASTR = 'CORRECTING PROFILE: SMOOTHING STEP VIA MLINE'
      CALL BANNER(ASTR)

      MFRT = 0.0
C...
C...
C...
      S = 0.0
      ST = 0.0
      J_DELTA = J_END - J_START + 1
c     write(6,*)' Number of points =',J_DELTA
C...put new data into the next I line
      I_LINE     = I_LINE + 1

C...start at top of nozzle and move down to centerline
      DO  J = J_START, J_END
C...find increment path lenght
       IF(J.NE.J_START)S = SQRT((Y(J) - Y(J-1))**2 + (X(J) - X(J-1))**2)
C...total path length
       ST = S + ST
      ENDDO
c     write(6,*)' Total path length =',ST

C...break up path in equal potions
      DS = ST / (J_DELTA*1.-1.)
c     WRITE(6,*)'DS = ',DS

      WRITE(6,1010)
      WRITE(I_FILE,*)' '
C...set the path length to 0
      S = 0

C... start spacing the data
      DO  J = J_START, J_END

C... increment the path length
       IF(J.NE.J_START)S = S + DS

C...fix the starting point
       IF(J.EQ.J_START)THEN
```

```fortran
         ZETA = 0.0
         J1   = J_START+1
         S    = 0.0
         GOTO 10
        ENDIF

C...fix the ending point
        IF(J.EQ.J_END)THEN
         ZETA = 1.0
         J1   = J_END
         S    = ST
         GOTO 10
        ENDIF


        ST1_OLD = ST
        ST1     = 0.0

C...find (X,Y) point on S path
        DO J1 = J_START+1, J_END

         S1= SQRT((Y(J1) - Y(J1-1))**2 + (X(J1) - X(J1-1))**2)
         ST1 = S1 + ST1

C...search for local path >= to S path
         IF(ST1.GE.S)THEN
          ZETA      = (S-ST1_OLD) / (ST1 - ST1_OLD)
c       write(6,*)'ZETA,S,ST1,ST1_OLD,J1',ZETA,S,ST1,ST1_OLD,J1
          GOTO 10
         ENDIF

         ST1_OLD = ST1

        ENDDO

        ZETA = 1.0

 10     CONTINUE
C...Since we advance by one point reset back
        JLO = J1-1
C...
C...
C...

C...Number of points
c      J_DELTA = J_END - J_START + 1
c      WRITE(6,*)'Number of points',J_DELTA
C...spaceing
c      DY     = Y(J_START) / (J_DELTA*1.-1.)
c      write(6,*)'spaceing =',DY
C...thickness
c      YY     = Y(J_START)
c      write(6,*)'thickness=',YY

c      I_LINE     = I_LINE + 1

c      WRITE(6,1010)
c      WRITE(I_FILE,*)' '
C...Evenly space the data
c      DO  J = J_START, J_END
c       IF(J.NE.J_START)YY = YY - DY
c       IF(YY.LT.0.0)YY=0.0

C... Find JLO
c       IF(J.EQ.J_START)THEN
c        JLO = 1
c       ELSEIF(J.EQ.J_END)THEN
c        JLO = J_END
c       ELSE
c        CALL HUNT(Y,J_END,YY,JLO)
c       ENDIF

C... Find Zeta
c       IF(JLO.GE.J_END)THEN
c        ZETA      = 0.0
c       ELSE
c        ZETA      = (YY-Y(JLO)) / (Y(JLO+1) - Y(JLO))
c       ENDIF

c       write(6,*)'YY,JLO,ZETA',YY,JLO,ZETA
```

```
C... Correct Zeta
c       IF(ZETA.LT.0.0)THEN
c        WRITE(6,*)'Warning: ZETA=',ZETA
c        ZETA = 0.0
c       ENDIF
c       IF(ZETA.GT.1.0)THEN
c        WRITE(6,*)'Warning: ZETA=',ZETA
c        WRITE(6,*)' JLO = ',JLO
c        ZETA = 1.0
c       ENDIF


        XY(I_LINE,J,1) = (1.-ZETA)*X(JLO) + ZETA*X(JLO+1)
        XY(I_LINE,J,2) = (1.-ZETA)*Y(JLO) + ZETA*Y(JLO+1)
        DO N = 1 , 8
         FLOW(I_LINE,J,N)   = (1.-ZETA)*FLOW(I_LINE-1,JLO,N) + ZETA*FLOW(I_LINE-1,JLO+1,N)
        ENDDO
c       WRITE(6,*)'X,YY,JLO,ZETA=',XY(I_LINE,J,1),YY,JLO,ZETA
       ENDDO

C...Reload the new data
       DO  J = J_START , J_END

       X(J) = XY(I_LINE,J,1)
       Y(J) = XY(I_LINE,J,2)
       P(J) = FLOW(I_LINE,J,6)
       R(J) = FLOW(I_LINE,J,7)
       U(J) = FLOW(I_LINE,J,1)
       V(J) = FLOW(I_LINE,J,2)
c      Q(J) = FLOW(I_LINE,J,4)
       Q(J) = SQRT(FLOW(I_LINE,J,1)**2 + FLOW(I_LINE,J,2)**2)
c      A(J) = FLOW(I_LINE,J,5)
       A(J) = ATAN(FLOW(I_LINE,J,2)/FLOW(I_LINE,J,1))

       X4 = X(J)
       Y4 = Y(J)
       P4 = P(J)
       R4 = R(J)
       Q4 = Q(J)
       A4 = A(J)
       U4 = U(J)
       V4 = V(J)
       CALL THERMO(Q4,P4,R4,T4,C,M4)

       XY(I_LINE-1,J,1)   = X(J)
       XY(I_LINE-1,J,2)   = Y(J)
       FLOW(I_LINE-1,J,1) = U(J)
       FLOW(I_LINE-1,J,2) = V(J)
       FLOW(I_LINE-1,J,3) = M4
       FLOW(I_LINE-1,J,4) = Q(J)
       FLOW(I_LINE-1,J,5) = A(J)
       FLOW(I_LINE-1,J,6) = P(J)
       FLOW(I_LINE-1,J,7) = R(J)
       FLOW(I_LINE-1,J,8) = T4

        IF(J.EQ.1)THEN
         AREA_X = 0.0
         AREA_Y = 0.0
         DEL_X  = 0.0
         DEL_Y  = 0.0
         R_AVE  = 0.0
         U_AVE  = 0.0
         V_AVE  = 0.0
        ELSE
         DEL_X  = ABS(X4-X(J-1))
         DEL_Y  = ABS((Y4**2 - Y(J-1)**2 ))*DELTA + (1-DELTA)*ABS(Y4-Y(J-1))
         AREA_X = PI * DEL_Y * DELTA + (1-DELTA)*DEL_Y
         AREA_Y = 2.0 * PI * (Y4 + Y(J-1))/2.0*DEL_X*DELTA + (1-DELTA)*DEL_X
         R_AVE  = (R4 + R(J-1))/2.0
         U_AVE  = (U4 + U(J-1))/2.0
         V_AVE  = (V4 + V(J-1))/2.0
        ENDIF

         MFR    = R_AVE*(U_AVE*AREA_X+V_AVE*AREA_Y)
         MFRT   = MFR + MFRT
         ERROR_M=(MFRT-MDOT)/MDOT

       AP = A4*RAD
       PP = P4 /GL

       IF (KW.EQ.1)THEN
```

```
        IF(J.EQ.1)THEN
         BACKSPACE(3)
         WRITE(3,1020)I_LINE,J,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
        ENDIF
        WRITE(6,1020)I_LINE-1,J+L_OFFSET,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
        WRITE(I_FILE,1020)I_LINE-1,J+L_OFFSET,X4,Y4,U4,V4,M4,Q4,AP,PP,R4,T4
        IF(J.EQ.J_END)write(6,*)' MASS FLUX = ',MFRT,
     >    ' PERCENT ERROR = ',ERROR_M*100.
       ENDIF

       ENDDO

 30    CONTINUE
       I_LINE     = I_LINE - 1


       RETURN
       END
       SUBROUTINE SPLINE(X,Y,N,YP1,YPN,Y2)
       DIMENSION X(N),Y(N),Y2(N),U(N)
       IF (YP1.GT..99E30) THEN
         Y2(1)=0.
         U(1)=0.
       ELSE
         Y2(1)=-0.5
         U(1)=(3./(X(2)-X(1)))*((Y(2)-Y(1))/(X(2)-X(1))-YP1)
       ENDIF
       DO 11 I=2,N-1
         SIG=(X(I)-X(I-1))/(X(I+1)-X(I-1))
         P=SIG*Y2(I-1)+2.
         Y2(I)=(SIG-1.)/P
         U(I)=(6.*((Y(I+1)-Y(I))/(X(I+1)-X(I))-(Y(I)-Y(I-1))
     *       /(X(I)-X(I-1)))/(X(I+1)-X(I-1))-SIG*U(I-1))/P
11     CONTINUE
       IF (YPN.GT..99E30) THEN
         QN=0.
         UN=0.
       ELSE
         QN=0.5
         UN=(3./(X(N)-X(N-1)))*(YPN-(Y(N)-Y(N-1))/(X(N)-X(N-1)))
       ENDIF
       Y2(N)=(UN-QN*U(N-1))/(QN*Y2(N-1)+1.)
       DO 12 K=N-1,1,-1
         Y2(K)=Y2(K)*Y2(K+1)+U(K)
12     CONTINUE
       RETURN
       END
```

INITIAL DISTRIBUTION LIST

		Copies
Weapon Systems Technology Information Analysis Center Alion Science and Technology 201 Mill Street Rome, NY 13440	Ms. Gina Nash gnash@alionscience.com	Electronic
Defense Technical Information Center 8725 John J. Kingman Rd., Suite 0944 Fort Belvoir, VA 22060-6218	Mr. Jack L. Rike jrike@dtic.mil	Electronic
AMSAM-L	Ms. Anne C. Lanteigne anne.lanteigne@us.army.mil	Electronic
	Mr. Michael K. Gray michael.k.gray@us.army.mil	Electronic
RDMR		Electronic
RDMR-CSI		Electronic
RDMR-SSM-A	Mr. Lamar Auman lamar.auman@us.army.mil	Electronic/Hardcopy
	Mr. Kevin Kennedy Kevin.kennedy4@us.army.mil	Electronic/Hardcopy
	Mr. Clark Mikkelson clark.mikkelson@us.army.mil	Electronic
Torch Technologies, Inc. 4035 Chris Drive SW Huntsville, AL 35802	Ms. Robin L. Ryan robin.l.ryan.ctr@us.army.mil	Electronic